Text to Phoneme Conversion in Persian Using Neural Networks

F. Hendessi
Electrical and Computer Engineering
Isfahan University of Technology

A . Ghayoori
Electrical and Computer Engineering
Isfahan University of Technology

Abstract: Speech is the most natural and widespread form of human communication. That's why speech synthesis has interested researchers for decades. In this paper, a Persian text to speech system is presented. The system uses speech waveform concatenation method that is comparatively mature in text-to-speech synthesis. This paper discusses the experimental study on the use of neural networks in text-to-speech systems for Persian language. In the context of text to phoneme conversion, the neural networks demonstrate good performance. It is shown that a network can capture significant portion of regularities in the Persian pronunciation as well as absorb many of the irregularities.

Keywords: Text to Speech, Neural Networks, Phoneme, HMM

1. Introduction

Voice communication has always been one of the most important areas of research in human-computer interaction. Although it was viewed as science fiction in the past, it is today a reality, with a lot of real-life applications. It is cleared that with the current level and rapid advancements of technology, working speech technology will be mainstream very soon. Speech synthesis systems have proven useful as a reading tool for the blind and have improved productivity for those involved in tasks where it is difficult to use a complete terminal. Other applications include emergency systems using synthetic speech to alert and direct people, data bank information retrieval using telephone [1]

Although the intelligibility of speech synthesizers is considered today adequate for many applications, synthesized speech sounds, in general, unnatural and can be easy recognized. For many years, linguistics has studied the speech production mechanism. The Object of this study is to examine the appropriateness of neural networks in an area of human-computer interaction, namely that of text-to-speech synthesis for the Persian language. In particular, neural network performance is examined for the text-to-phonemes conversion phase. Especially for the Persian language, the research in this area is very limited. Similar systems have been presented for English [2] [3].

In designing a synthesizer, the English language has some simplifying advantages over Persian that makes the synthesizing process much easier. In English literature, the vowels are

written in addition to consonants in a word, while in Persian only long vowels are written and short vowels are omitted.
Unfortunately, since some letters can appear both as a consonant and a vowel (long vowel) in Persian, even distinguishing the long vowels is not an easy task. So the first and the most important step in designing a Persian synthesizer is to vowelize each word in a text.
There are several ways in which Persian words can be translated into phonemic spelling. The first and the most common one is to assign each word, its phonemes and store these assignments in a lexicon. In this scheme when the algorithm encounters a word, it begins to search its database to find the corresponding vowels of the queried word. This is known as dictionary method [1]. Several commercial methods for text to phoneme conversion rely on an exhaustive dictionary approach. However, not all words are included in the dictionaries and therefore the phoneme recognition performance is compromised. One common approach to mitigate this problem is to make a database searching for the best match for requested word.
The other way (which is proposed by the authors) is to train an NN for extracting the structure of a vowelization process in Persian. For this purpose a training data should be gathered that properly cover the contemporary language. Applying neural network for phoneme mapping in text-to-speech conversion creates a fast distributed recognition engine, which supports the mapping of missing words in the database.
In this paper we develop a method with the following stages: *1)* Text-to-Phoneme Conversion, *2)* Speech generation

2. Text-to-Phoneme Conversion

An important stage in designing a Persian synthesizer is to insert short vowels along with consonants and long vowels in a word. The short vowels are inserted as diacritic marks, which are placed above or under consonants.
As mentioned before, a solution to vowelization process of the Persian text is a system to learn pronunciation rules. Such an intelligent system will be able to obtain some knowledge and to preserve this knowledge. Neural networks are perfect candidates for such systems. They have been used in many aspects of speech processing, where they have demonstrated very good performance [4]. The neural system receives, as input, the letter to be pronounced and all other information that affect the pronunciation of that letter (such as surrounding letters). The output of the system could be the phoneme that must be pronounced.
The system described above have some advantages. In this system the processing can be done in parallel due to the structure of the neural networks. This is very helpful because the neural network structure can be implemented in hardware and then the processing can be done in real time, a factor that is very important in voice communication. The other advantages is that by giving by giving examples of words and their pronunciation to the network, the system will be able to find association between the words and their phonemes. It will therefore be able to generalize and obtain some outputs even when presented with previously unseen words.
In this paper, the vowel recognition is done, using two layers Multi Perceptron neural network (MLP), which is build on the pioneering work of Sejnowski and Rosenberg (i.e., NetTalk [3]) (Fig. 1). The NN is hierarchically arranged into three layers of unit: an input layer, an output layer and an intermediate or hidden layer. Information flows through the network from input to output. Each unit in the hidden layer receives inputs from all of the input units of the previous layer, and in turn sends its outputs to the all the units in the output layers.

There are many learning algorithms for training the NN by which desirable weights of the NN are achieved. One of the most common algorithms that are used is Back Propagation.

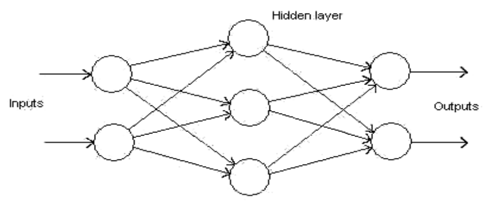

Fig.1: MLP neural network

The learning rate in Back propagation algorithm is chosen to be a 3-valued function with respect to the error of the output nodes.

2.1. Preprocessing:

For reducing the amount of information, which is required to be learned by the NN, some preprocessing could be accomplished. Because of the complexity of the vowelization process in Persian, apply some rules increase the efficiency of the process.
As an example of these rules; there some combinations of letters of letter appear at special positions in a word, the ambiguity about the vowel of them vanished and it can be determined simply by looking up at a rule lexicon.
Some of the words in Arabic are widely used in Persian. Arabic is a rule-based language in which the vowelization process obeys some known rules. So to vowelize these inserted words, Arabic rules are stored along with other extracted rules For Persian, in a database
There are some words, which the extracted rules cannot be applied to. So these words and their corresponding phonemic structures should be stored in a dictionary. So the fist step in preprocessing stage is to check the dictionary to separate these exceptional words. As mentioned before applying thes rules increase the efficiency of the vowelization process.
Other part of the preprocessing is to separate the suffix and/or prefix of each word in the text. There are many suffixes and prefixes in Persian language that should be separated from the original word before entering the word into the NN. These suffixes and prefixes result in many different meanings from one word. Consequently by learning a main word, the NN can correctly vowelize all of its variations.

2.2. Training the NN

A learning set is prepared which contains the common words in Persian. The words in this set were completely vowelized. Before the NN begins to process the letters, each letter of the word is mapped into a 6-bit string (There are 32 alphabets and some additional signs in Persian which should enter the NN as inputs, so six bits seem sufficient for mapping all of these different inputs). The complete list of this mapping is depicted in table 1.
The six bits equivalence of each letter in a word are entered into the neural network as inputs with four adjacent right and four adjacent left letters. For the letters that are located near the ends of the words, blanks are placed in the empty positions. We consider a sliding window,

similar to NetTalk, with the letter to be vowelized placed in the middle of the window (i.e. Central Window Positioning [5]).

Table 1: The lookup table used for entering alphabets into NN

000001	ا	010011	ط
000010	ب	010100	ظ
000011	پ	010101	ع
000100	ت	010110	غ
000101	ث	010111	ف
000110	ج	011000	ق
000111	چ	011001	ک
001000	ح	011010	گ
001001	خ	011011	ل
001010	د	011100	م
001011	ذ	011101	ن
001100	ر	011110	و
001101	ز	011111	ه
001110	ژ	100000	ی
001111	س	100001	آ
010000	ش	100010	ّ
010001	ص		
010010	ض		

In alignment approach presented in [3] the window positioning structure is changed to Second Position Asymmetric Windowing (SPAW), where the number of spaces before and after the object letter is not equal. According to [3], this structure is called "Second Position" because the object letter in the second space from the middle of the central window. Second position asymmetric windowing is labeled "N-M SPAW" for notation purposes, where N & M refer to the number of spaces before and after the central window position. This structure is completely explained in [5].
Four adjacent letters at each side is considered to enter into the NN as input in addition to the letter which is under process. This amount of adjacent letters is the minimum requirement for determining the vowel of the letter without ambiguity in Persian. If less adjacent letters are considered, there would be some words that for the similar input of the NN the output would be different.
There are some signs that are used in Persian (e.g. "tashdid" (germination, "tanween",..) and NN have a mechanism to handle these cases. The output of the neural network shows the vowel of the letter .As mentioned before; a letter can have a short vowel or a long vowel. So the output of the NN should have the ability to show these vowels in addition to the case that the letter under process doesn't have any vowel. The outputs of the NN also should help us to determine some exceptions (we talk about this in future). To distinguish long vowels, NN act as follows. If a letter was followed by one of these special letters "ا "and "و "and "ی", it can be

candidate to have long vowel. The final deduction is based upon the fact that whether those three letters doesn’t appear as consonant. Unfortunately these three letters sometimes appear as consonants and in this case, they wouldn’t be the indicator of long vowel for their previous letter (Fig.2). Distinguishing these cases is the task of NN.
If the letters ا and و and ی were the indicator of long vowels of their previous letter, NN generates special codes as an output to indicate the presence of long vowels. In this case the NN doesn’t do any processing for these three letters (ا,و ,ی) and jump over them.
The output of the NN is also used to indicate some exceptions. There are some letters (in a special order) in Persian with an orthography that does not correspond to their phonetic structure (e.g. خوا), but only in some special words, as otherwise they are pronounced regularly.

```
                         توجه (/tævæjɔ:/)
Input=bbbbbb bbbbbb bbbbbb bbbbbb 000100  011110 000110 011111bbbbbb   Output=011
Input=bbbbbb bbbbbb bbbbbb 000100 011110 000110 011111 bbbbbb bbbbbb   Output=011
Input=bbbbbb bbbbbb 000100 011110 000110 011111 bbbbbb bbbbbb bbbbbb   Output=010
                           توپ (/tu:p/)
Input=bbbbbb bbbbbb bbbbbb bbbbbb 000100 011110 000011 bbbbbb bbbbbb   Output=100
```

Fig. 2: Consider these two words (toop) and (tavajoh). In the first one the output of the NN for the first letter is an indicator of a long vowel but in the second one the output of the NN would be a short vowel.

When the NN encounters these exceptions, it alerts the application (fifth case) and the application labels these exceptions so they can be pronounced correctly. For example consider the two words “خواهر” (/xa:hær/) and “خواص”(/xæva:s/). When the NN wants to specify the vowel of the letter “خ” in the first word, it generates 011(indicating the short vowel /æ/) as an output. However, for the first letter of the second word, the NN alerts the application about the presence of an exception. When the application receives this message, it understands that in the second word, the 2nd and 3rd letters indicate a long vowel “ا” and treats them properly. The complete list of NN’s output is shown in table 2.

Table 2: the outputs of neural net

001	ـِ	/e/
010	ـُ	/u/
011	ـَ	/æ/
100	Long vowel indicator	
101	Special case indicator	
110	“Sokoon”	

3.Unknown Text

When the NN is trained once, it can be used in text-to-phoneme part of an speech synthesizer. The trained neural net processes the text sequentially on a word-by-word basis. In this stage, first the prefixes and suffixes are separated from each word, and then the NN processes each word, letter by letter. While the NN moves from one letter to another, the extracted Persian rules (which are mentioned in section 2) are checked. If one of the rules is applicable, the NN ignores the corresponding letters. These rules decrease the amount of processing done by the NN in both the training and testing stages. They also increase the accuracy of synthesizing for unknown words, which neural net isn't trained for. By moving the neural net, one string of vowels is produced that should be located properly between letters of the word. For the unknown words, the outputs of NN may generate a final diacritized word, which is unpronounceable. To fix these difficulties some post processing are also required.

4. Experimental Results

Various neural network structures have been examined, in order to select the appropriate structure to be used. The same training set was tested on different architectures with different numbers of hidden units. Based on the preliminary experimental results, a neural network with 200 up to 300 hidden neurons seemed to be the most appropriate structure .
The purpose of the training was to minimize the mean square error and consequently the percentage of incorrectly classified input patterns. Another important goal is to avoid over training, which could minimize further the error on the testing set, but at the same time increase the error on the testing set. After testing the NN for a text of 800words and evaluating the system by measuring the intelligibility of the synthesis speech the performance of 80-88% is achieved. The intelligibility level is judged by 10 Persian native language listeners and is illustrated in table 3.

Table 3: Understanding results of a closed evaluation test

	A	B	A+B (Intelligibility percentage)	C
Listener 1	80.1%	7.6%	87.7%	12.3%
Listener 2	78.75%	5.25%	84%	16%
Listener 3	85.1%	8.4%	93.5%	6.5%
Listener 4	80.7%	3.5%	84.2%	15.8%
Listener 5	83.2%	5.1%	88.3%	11.7%
Listener 6	81.2%	5.3%	86.5%	13.5%
Listener 7	79.2%	5.3%	84.5%	15.5%
Listener 8	82.3%	3.1%	85.4%	14.6%
Listener 9	81%	5.0%	86%	14%
Listener 10	80.1%	5.3%	85.4%	14.6%
Average	81.2%	5.4%	86.6%	13.4%

Further improvements can be achieved by using probabilistic models to ameliorate some of the errors, which is occurred on the testing set [6].

Considering the initial results and the room for future improvement, a satisfactory system for Persian text-to-speech conversion could be developed for a number of real-life applications

References:

[1] Cohen, P.R. and S.L. Oviatt, *The Role of Voice in Human-Machine communication, Voice Communication between Humans and Machines,* National Academic Press, DC, 1994

[2] Sejnowski, T.J. and C.R. Rosenberg, NETTalk: A Parallel network that learns to read aloud, *Electrical Engineering and Computer Science Technical Report JHU/EECS-86/01*, Johns Hopkins University, Baltimore, 1986

[3] Sejnowski, T.J. and C.R. Rosenberg, Parallel networks that learn to pronounce English text, *Complex Systems*, vol.1, 145-168, 1987

[4] Morgan, D.P. and C.L. Scofield, Neural Networks and Speech Processing, *Kluwer Academic Publishers*, 1991.

[5] Mark J. Embrechts and /Fabio Arciniegas, Neural Networks for Text-to-Speech Phoneme Recognition, , *IEEE International Conference on Systems, Man, and Cybernetics,* Oct. 2000, 3582 - 3587 vol.5

[6] A. Ghayoori, A Complete Text-to-Speech system for Persian, *Master's Thesis*, 2000