Comparing Two Models for Word Boundary Detection in a Phoneme Sequence Using Recurrent Fuzzy Neural Networks

Mohammad Reza Feizi Derakhshi [1, 2], Mohammad Reza Kangavari [2],
Elnaz Zafarani Moattar [3], Mahdi Aliyari Shorehdeli [4]
1 Computer Science faculty, University of Tabriz
2 Computer engineering department, University of science and technology of Iran
3 Islamic Azad University Science and Research Branch
4 K.N. Toosi University of Technology
{m_feizi, kangavari}@iust.ac.ir, elnazzafarani@yahoo.com, Aliyari@gmail.com

Abstract

The word boundary detection has an application in speech processing systems. The problem this paper tries to solve is to separate words of a sequence of phonemes where there is no delimiter between phonemes. This paper tries to compare two conceptual models for word boundary detection, named preorder and postorder models. In this paper, at first, a recurrent fuzzy neural network (RFNN) together with its relevant structure is proposed and learning algorithm is presented. Next, this RFNN is used to post-detecting word boundaries. Some experiments have already been implemented to determine complete structure of RFNN in both models. Here in this paper, three methods are proposed to encode input phoneme and their performance have been evaluated. Some experiments have been conducted to determine required number of fuzzy rules and then performance of RFNN in postdetecting word boundaries is tested in both models. Preorder model shows better performance than postorder model in experiments.

Key words: *Word boundary detection, Recurrent fuzzy neural network, Fuzzy neural network, Fuzzy logic, Natural language processing, Speech processing*

1. Introduction

In this paper an attempt is made to compare two conceptual models named preorder model and postorder model by using Recurrent Fuzzy Neural Network (RFNN). The problem under this study is word boundary detection. This needs a required number of delimiters that should be inserted into the given sequence of phonemes. In so doing, the essential step to be taken is to detect word boundaries. This is the place where a delimiter should be inserted. The word boundary detection has an application in speech processing systems, where a speech recognition system generates a sequence of phonemes which form speech. It is necessary to separate words of the generated sequence before going through further phases of speech processing.

Figure 1 illustrates general model for continuous speech recognition systems [13], [14]. As we can see, at first a preprocessing occurs to extract features. Output of this phase is feature vectors which are sent to phoneme recognition (acoustic decoder) phase. In this phase feature vectors are converted to phoneme sequence. Later, the phoneme sequence enters the "phoneme to word decoder block" where it is converted to word sequence [13]. Finally word sequence is delivered to linguistic decoder phase which tests grammatical correctness [13], [15]. The problem under this study is placed in a phoneme to word decoder.

In most of current speech recognition systems, phoneme to word decoding is done by using word database. Within these systems, the word database is stored in structures such as lexical tree or Markov model. However, in this study the attempt is made to look for an alternative method that is able to do decoding without using word database. Although using word database can reduce error rate, it is useful to be independent from word database in some applications; when, for instance, a small number of words in a great volume of speech with large vocabulary (e.g. news) is sought. In such application, it is not economical to make a system with large vocabulary to search for only small number of words. It should be noted that it is necessary to make a model for each word. It is not only unaffordable to construct these models, but the large number of models also require a lot of run time to make a search. While making use of a system which can separate words independent of word database, appears to be very useful since it is possible to make word models of a small number of words and to avoid from unnecessary complications. However, these word models are still needed with a difference that search in word models can be postponed

978-1-4244-1841-1/08/$25.00 ©2008 IEEE

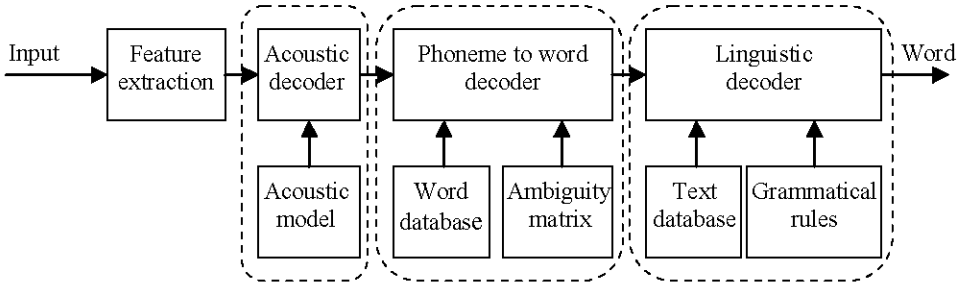

Figure 1: General model for continuous speech recognition systems. [13]

to next phase where word boundaries determined with some uncertainty. Thus, the word which is looked for can be found faster and with a less cost.

Previous works on English language have low performance in this field [8] whereas the works on Persian language seems to have acceptable performance, for there are structural differences between Persian and English language system [5]. (See section 2 for some differences in syllable patterns) Our [1], [2], [3], [4] and others [13] previous works confirm this.

In general, the system should detect boundaries considering all phonemes of input sequence. In order to make the work simple, the problem usually is reduced to making decision about existence of boundary after current phoneme given the previous phonemes. But in this work we propose a different idea as making decision about existence of boundary before current phoneme given phonemes after the boundary. Since the system should predict existence of boundary before current phoneme, it is a time series prediction problem.

Lee and Teng in [6] used five examples to show that RFNN is able to solve time series prediction problems. In the first example, they solve simple sequence prediction problem found in [8]. Second, they solve a problem from [11] in which the current output of the planet is a nonlinear transform of its inputs and outputs with multiple time delay. As third example, they test a chaotic system from [10]. They consider in forth example the problem of controlling a nonlinear system which was considered in [11]. Finally, the model reference control problem for a nonlinear system with linear input from [12] is considered as fifth example. Our goal in this paper is to evaluate the performance of RFNN in word boundary detection.

The paper is organized as follows. Section 22 and 53 present RFNN structure and the learning algorithm respectively. Experiments and their results are presented in section 4. Section 5 is given to the conclusion of this paper.

2. Structure of the Recurrent Fuzzy Neural Network (RFNN)

Ching-Hung Lee and Ching-Cheng Teng introduced a 4 layered RFNN in [6]. We used that network in this paper. Figure 2 illustrates the configuration of the proposed RFNN. This network consists of n input variables, $m \times n$ membership nodes (m-term nodes for each input variable), m rule nodes, and p output nodes. Therefore, RFNN consists of $n + mn + m + p$ nodes, where n denotes number of inputs, m denotes the number of rules and p denotes the number of outputs.

2.1. Layered Operation of the RFNN

This section presents operation of nodes in each layer. In the following description, u_i^k denotes the ith input of a node in the kth layer; O_i^k denotes the ith node output in layer k.

Layer 1: Input Layer: Nodes of this layer are designed to accept input variables. So output of these nodes is the same as their input, i.e.,

$$O_i^1 = u_i^1 \tag{1}$$

Layer 2: Membership Layer: In this layer, each node has two tasks simultaneously. First it performs a membership function and second it acts as a unit of memory. The Gaussian function is adopted here as a membership function. Thus, we have

$$O_{ij}^2 = \exp\left\{-\frac{(u_{ij}^2 - m_{ij})^2}{(\sigma_{ij})^2}\right\} \tag{2}$$

where m_{ij} and σ_{ij} are the center (or mean) and the width (or standard deviation) of the Gaussian membership function. The subscript ij indicates the jth term of the ith input x_i. In addition, the inputs of this layer for discrete time k can be denoted by

$$u_{ij}^2(k) = O_i^1(k) + O_{ij}^f(k) \tag{3}$$

where

$$O_{ij}^{f}(k) = O_{ij}^{2}(k-1).\theta_{ij} \quad (4)$$

and θ_{ij} denotes the link weight of the feedback unit. It is clear that the input of this layer contains the memory terms $O_{ij}^{2}(k-1)$, which store the past information of the network. Each node in this layer has three adjustable parameters: m_{ij}, σ_{ij}, and θ_{ij}.

Layer 3: Rule Layer: The nodes in this layer are called rule nodes. The following AND operation is applied to each rule node to integrate these fan-in values, i.e.,

$$O_i^3 = \prod_j u_{ij}^3 \quad (5)$$

The output O_i^3 of a rule node represents the "firing strength" of its corresponding rule.

Layer 4: Output Layer: Each node in this layer is called an output linguistic node. This layer performs the defuzzification operation. The node output is a linear combination of the consequences obtained from each rule. That is

$$y_j = O_j^4 = \sum_{i=1}^{m} u_i^4 \ w_{ij}^4 \quad (6)$$

where $u_i^4 = O_i^3$ and w_{ij}^4 (the link weight) is the output action strength of the jth output associated with the ith rule. The w_{ij}^4 are the tuning factors of this layer.

2.2. Fuzzy Inference

A fuzzy inference rule can be proposed as

$$y_j = O_j^4 = \sum_{i=1}^{m} u_i^4 \ w_{ij}^4 \quad (7)$$

RFNN network tries to implement such rules with its layers. But there is some difference! RFNN implements the rules in this way:

$$\begin{aligned} R^j: \quad & IF \ \ u_{1j} \ \ is \ \ A_{1j}, \ldots, u_{nj} \ \ is \ \ A_{nj} \\ & THEN \ \ y_1 \ \ is \ \ B_1^j \ \ldots y_P \ \ is \ \ B_P^j \end{aligned} \quad (8)$$

where $u_{ij} = x_i + O_{ij}^{2}(k-1).\theta_{ij}$ in which $O_{ij}^{2}(k-1)$ denotes output of second layer in previous level and θ_{ij} denotes the link weight of the feedback unit. That is, the input of each membership function is the network input x_i plus the temporal term $O_{ij}^{2}\ \theta_{ij}$. This fuzzy system, with its memory terms (feedback units), can be considered as a dynamic fuzzy inference system and the inferred value is given by

$$y^* = \sum_{j=1}^{m} \alpha_j \ w_j \quad (9)$$

where $\alpha_j = \prod_{i=1}^{n} \mu_{A_{ij}}(u_{ij})$. From the above description, it is clear that the RFNN is a fuzzy logic system with memory elements.

3. Learning Algorithm for the Network

Learning goal is to minimize following cost function:

$$E(k) = \frac{1}{2}\sum_{i=1}^{P}(y_i(k) - \hat{y}_i(k))^2 = \frac{1}{2}\sum_{i=1}^{P}(y_i(k) - O_i^4(k))^2 \quad (10)$$

where $y(k)$ is the desired output and $\hat{y}(k) = O^4(k)$ is the current output for each discrete time k.

Well known error back propagation (EBP) algorithm is used to train the network. EBP algorithm can be written briefly as:

$$W(k+1) = W(K) + \Delta W(k) = W(k) + \eta\left(-\frac{\partial E(k)}{\partial W}\right) \quad (11)$$

where W represents tuning parameters and η is the learning rate. As we know, tuning parameters of the RFNN are m, σ, θ and w. By applying the chain rule recursively, partial derivation of error with respect to above parameters can be calculated.

4. Experiments and Results

Here, we want to define our preorder and postorder conceptual models. Consider a sequence of phonemes like $P_1P_2P_3P_4$. There may be some boundaries between them. It can be supposed to be between P_2 and P_3 i.e. $P_1P_2bP_3P_4$. In postorder [1] model, phonemes before boundary (P_1P_2) is used for detecting boundary and in postorder[2] model, phonemes before boundary (P_1P_2) is used for detecting boundary.

As it is mentioned, this paper tries to solve word boundary detection problem. System input is phoneme sequence and output is existence of word boundary before/after current phoneme. Because of memory element in RFNN, there is no need to hold previous phoneme in its input. So, the input of RFNN is a phoneme in the sequence and the output is the existence of boundary before this phoneme.

We used supervised learning as RFNN learning method. A native speaker of Persian language is used to produce a training set to train RFNN. He is supposed to

[1] We call it postorder because in this model boundary takes place after phonemes (for example: P_1P_2b)

[2] We call it preorder because in this model boundary takes place before phonemes (for example: bP_3P_4)

Table 1. Training time (T. time) and MSE error for different number of epochs for each coding method for both of models. (h: hour, m: minute, s: second)

Encoding method	Num. of epochs	Preorder model		Postorder model	
		T. time	MSE	T. time	MSE
Real	2	3.52 s	0.5454	3.66 s	0.60876
Real	20	32.03 s	0.5185	32.42 s	0.59270
Real	200	299.35 s	0.5207	312.61 s	0.58830
1 / 29	2	23.04 m	1	22 m	1
1 / 29	20	1 h, 32	1	1 h, 16 m	1
Binary	2	11.10 s	0.4678	11.50 s	0.55689
Binary	20	102.38 s	0.3904	102.39 s	0.51020
Binary	200	4.99 m	0.3103	17 m	0.46677
Binary	1000	25.10 m	0.2986	1 h, 24 m	0.45816

determine word boundaries and marked them. The same process was done for test set but boundaries were hidden from the system.

Each of test and training sets consists of about 15000 phonemes from daily speeches in library environment.

As it is mentioned, network input is a phoneme but this phoneme should be encoded before any other process. Thus, to encode 29 phonemes [16] in standard Persian, three methods for phoneme encoding were used in our experiments. These are as follow:

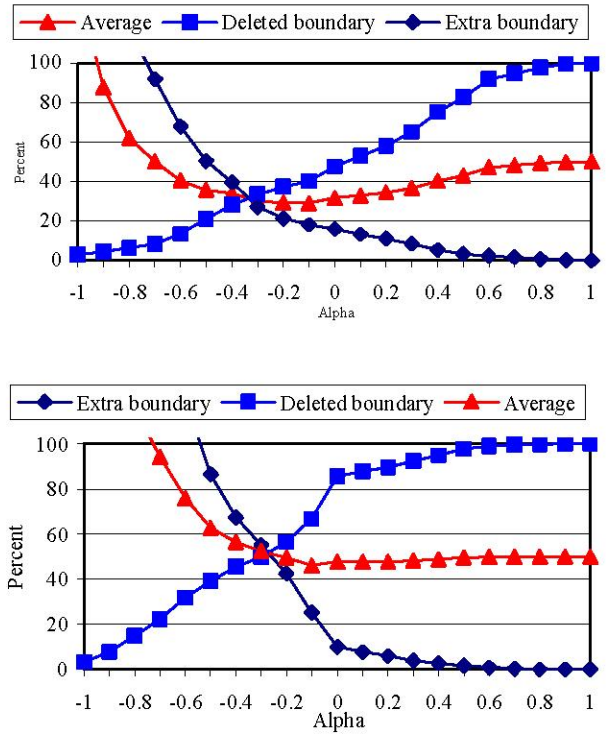

Figure 2: Extra boundary (boundary in network output, not in test set), deleted boundary (boundary not in network output, but in test set) and average error for different values of α. Up: preorder model, Down: postorder model.

Table 2. Some of experimental results to determine number of fuzzy rules for both of models (Training with 100 and 500 Epochs).

Num of rules	MSE (Preorder model)		MSE (Postorder model)	
	100 Epoch	500 Epoch	100 Epoch	500 Epoch
10	0.57068	0.48487	0.5452	0.5431
20	0.41413	0.41528	0.5455	0.5449
30	0.36787	0.35753	0.5347	0.5123
40	0.40841	0.39218	0.5339	0.5327
50	0.39282	0.38819	0.4965	0.4957
60	0.34826	0.33425	0.4968	0.4861
70	0.38438	0.37824	0.4883	0.4703
75	0.37100	0.35042	0.5205	0.5078
80	0.32822	0.30921	0.5134	0.4881
85	0.32637	0.31796	0.4971	0.4772
90	0.35164	0.32902	0.5111	0.4918
100	0.34639	0.33031	0.4745	0.4543

1. Real coding: In this method, each phoneme mapped to a real number in the range [0, 1]. In this case, network input is a real number.

2. 1-of-the-29 coding: In this method, for each input phoneme we consider 29 inputs corresponding to 29 phonemes of Persian. At any time only one of these 29 inputs will set to one while others will set to zero. Therefore, in this method, network input consists of 29 bits.

3. Binary coding: In this method, ASCII code of phoneme used for phonetic transcription, is transformed to binary and then is fed into network inputs. Since only lower half of ASCII characters are used for transcription, 7 bits are sufficient for this representation. Thus, in this method network inputs consists of 7 bits.

Some experiments have been implemented to determine performance of above mentioned methods. Table 1 shows some of the results. Obviously, 1-of-the-29 coding is not only time consuming but also it yields a poor result. When comparing binary with real coding, it is shown that although real coding requires less training time, it has less performance. It is not the case with binary coding. Therefore, binary coding method should be selected for the network. So far, 7 bits for input and 1 bit for output has been confirmed, so, to determine complete structure of the network, the number of rules has to be determined.

The results of some experiments with different number of rules and epochs are presented in Table 2. The best performance considering training time and mean squared error (MSE) is obtained in 80 rules for preorder model and in 60 rules for postorder model.

Table 3. Comparison of results.

Reference num	[8]	[1]	[2]	[3]	[4]	[13]
Error rate (Percent)	55.3	23.71	29.14	45.96	36.60	34

Although in some cases increasing in rule numbers results in a decrease in MSE, this decrease in MSE is not worth network complication. However, over train problem should not be neglected.

Now, the RFNN structure is completely determined: 7 inputs, 80/60 rules and one output. So, the main experiment for determining performance of the RFNN in this problem has been done. The RFNN is trained with the training set; then is tested with test set. Network determined outputs are compared with oracle determined outputs. The RFNN output is a real number in the range [-1, 1]. A hardlim function as follow is used to convert its output to a zero-one output.

$$T_i = \begin{cases} 1 & \text{if } O_i >= \alpha \\ 0 & \text{if } O_i < \alpha \end{cases} \qquad (12)$$

where α is a predefined value and O_i determines ith output of network. Value one for Ti means existence of boundary and vice versa. Boundaries for different values of α are compared with oracle defined boundaries. Results are presented in Figure 3. It can be seen that the best result is produced when α = -0.1 with average error rate 29.14%/45.96%.

5. Conclusions

In this paper a Recurrent Fuzzy Neural Network was used for word boundary detection. Three methods are proposed for coding input phoneme: real coding, 1-of-the-29 coding and binary coding. The best performance in experimental results, were achieved when the binary coding had been used to code input. The optimum rules number was 80/60 rules as well.

After completing network structure, experimental results showed average error 29.14%/45.96% on test set which is a good performance in compare with previous works [1], [4], [8], [13]. Table 3 presents error percentage of each reference. As it is seen, works on English language ([8]) have higher error than Persian language ([1], [3], [4], [13]). It seems that the difference is because of simple structure of Persian language. Also it can be seen that preorder model is better than postorder model. It shows that phonemes after boundary have more information about that boundary than phonemes before boundary. Postorder model is similar in concept with current speech recognition systems. Therefore, speech recognition systems should pay more attention to phonemes after boundary.

References

[1] M. R. Feizi D., M. R. Kangavari, "Preorder fuzzy method for determining word boundaries in a sequence of phonemes", 6th Iranian Conference on Fuzzy Systems and 1st Islamic World Conference on Fuzzy Systems, 2006.

[2] M. R. Feizi D., M. R. Kangavari, "Word Boundary Detection in a Phoneme Sequence in Persian Language Using Recurrent Fuzzy Neural Networks", Int. Conf. on Life System Modeling and Simulation (LSMS2007), 14 – 17 Sep. 2007, Also in DCDIS, 2007.

[3] M. R. Feizi D., M. R. Kangavari, "Using Recurrent Fuzzy Neural Networks for Predicting Word Boundaries in a Phoneme Sequence in Persian Language", 12th Int. Conf. on Human-Computer Interaction (HCII2007), 22 – 27 July 2007, Also in LNCS, Springer, 2007.

[4] M. R. Feizi D., M. R. Kangavari, "Inorder fuzzy method for determining word boundaries in a sequence of phonemes", 7thConf. on Intelligence Systems (CIS2005), 2005

[5] M. R. Feizi D., "Study of role and effects of linguistic knowledge in speech recognition", 3rd conference on computer science and engineering, 2000.

[6] C.-H. Lee, C.-C. Teng, "Identification and control of dynamic systems using recurrent fuzzy neural networks", IEEE Transactions on Fuzzy Systems, vol. 8, no. 4, pp.349-366, Aug. 2000.

[7] Y. Zhou, Sh. Li, R. Jin, "A new fuzzy neural network with fast learning algorithm and guaranteed stability for manufacturing process control", Elsevier, Fuzzy sets and systems 132, pp. 201-216, 2002.

[8] J. Harrington, G. Watson, M. Cooper, "Word boundary identification from phoneme sequence constraints in automatic continuous speech recognition", 12th conference on Computational linguistics, August 1988.

[9] S. Santini, A. D. Bimbo, R. Jain, "Block-structured recurrent neural networks," Neural Networks, vol. 8, no. 1, pp. 135–147, 1995.

[10] G. Chen, Y. Chen, H. Ogmen, "Identifying chaotic system via a wiener-type cascade model", IEEE Transaction on Control Systems, pp. 29–36, Oct. 1997.

[11] K. S. Narendra, K. Parthasarathy, "Identification and control of dynamical system using neural networks", IEEE Trans. on Neural Networks, vol. 1, pp. 4–27, Jan. 1990.

[12] C. C. Ku, K. Y. Lee, "Diagonal recurrent neural networks for dynamic systems control," IEEE Transaction on Neural Networks, vol. 6, pp. 144–156, Jan. 1995.

[13] B. Babaali, M. Bagheri, Kh. Hosseinzade, M. Bahrani, H. Sameti, "A phoneme to word decoder based on vocabulary tree for Persian continuous speech recognition", Int. Annual Computer Society of Iran Computer Conf., 2004.

[14] I. Gholampoor, "Speaker independent Persian phoneme recognition in continuous speech", PhD thesis, Electrical Engineering Faculty, Sharif University of Tech., 2000.

[15] N. Deshmukh, A. Ganapathiraju, J. Picone, "Hierarchical Search for Large Vocabulary Conversational Speech Recognition", IEEE Signal Processing Magazine, Vol. 16, No. 5, pp.84-107, September 1999.

[16] A. Najafi, "Basics of linguistics and its application in Persian language", Nilufar Publication, 1992.