Journal of AI and Data Mining
Vol 7, No 1, 2019, 137-147
DOI: 10.22044/JADM.2018.6136.1725

Improving Phoneme Sequence Recognition using Phoneme Duration Information in DNN-HSMM

M. Asadolahzade Kermanshahi and M.M. Homayounpour*

Computer Engineering and IT Department, Amirkabir University of Technology, Tehran, Iran

Received 17 August 2017; Revised 23 February 2018; Accepted 19 May 2018
**Corresponding author: homayoun@aut.ac.ir (M.M. Homayounpour).*

Abstract
Improving phoneme recognition has attracted the attention of many researchers due to its applications in various fields of speech processing. The recent research achievements show that using deep neural network (DNN) in speech recognition systems significantly improves the performance of these systems. There are two phases in the DNN-based phoneme recognition systems including training and testing. Most previous research works have attempted to improve training phases such as training algorithms, different types of network, network architecture and feature type. However, in this work, we focus on the test phase, which is related to the generation of phoneme sequence that is also essential to achieve a good phoneme recognition accuracy. Past research works have used Viterbi algorithm on hidden Markov model (HMM) to generate phoneme sequences. We address an important problem associated with this method. In order to deal with the problem of considering geometric distribution of state duration in HMM, we use real duration probability distribution for each phoneme with the aid of hidden semi-Markov model (HSMM). We also represent each phoneme with only one state to simply use phoneme duration information in HSMM. Furthermore, we investigate the performance of a post-processing method that corrects the phoneme sequence obtained from the neural network based on our knowledge about phonemes. The experimental results obtained using the Persian FarsDat corpus show that using the extended Viterbi algorithm on HSMM achieves phoneme recognition accuracy improvements of 2.68% and 0.56% over the conventional methods using Gaussian mixture model-hidden Markov models (GMM-HMMs) and Viterbi on HMM, respectively. The post-processing method also increases the accuracy compared to before its application.

Keywords: *Phoneme Recognition, Deep Neural Network, Hidden Markov Model, Hidden Semi-Markov Model, Extended Viterbi Algorithm, Phoneme Duration, Persian (Farsi) Language.*

1. Introduction

The role of a phoneme recognition system is to identify the phoneme sequence in the input speech signal. A phoneme recognizer can be used in many applications such as language identification, speech summarization, spoken term detection, spoken document retrieval and topic identification [1-4]. Therefore, the performance of these systems is highly related to the performance of phoneme recognizers.

In the literature of speech recognition systems, speech recognition systems were first based on the Gaussian mixture model-hidden Markov model (GMM-HMM). Then neural network-hidden Markov models (NN-HMMs) were introduced to improve the performance of GMM-HMMs by predicting the posterior probabilities of states using observations [5-8]. NNs can model various speaking styles and background conditions [9]. The neural networks used in the NN-HMM models had first a few number of hidden layers and hidden units. However, since 2006, some techniques were introduced to train deep neural networks (DNNs), and more advances were achieved in computer hardware and their computational performance. Due to these advances, using neural network with multiple layers became more feasible. In [5, 10, 11], neural networks with multiple layers and a large number of hidden units were investigated, and the research works showed that deep neural networks were

more suitable for speech recognition than the shallower ones. Also, using deep neural networks in phoneme recognition led to a better performance compared to the previous models such as GMM-HMM [6].

In this paper, we aim to use deep neural networks for phoneme recognition in the Persian language. For this language, there is a research work on a DNN-based speech recognition system in [12], where the aim was speaker adaptation for this task. They proposed two speaker adaptation methods to adapt a system with speaker variability. In the first method, the input feature vectors of each speaker are normalized non-linearly in forward-backward passes. In the second one, DNN is changed to be able to adapt itself dynamically. Their results showed that the two proposed adapted DNN models improved phoneme recognition accuracy of the speaker-independent model.

The accuracy of a deep neural network-based phoneme recognition system depends on the accuracy of training neural network and using trained neural network in the test phase to find the phoneme sequence. In order to train neural network, a feature vector is extracted from each frame, and windows of frames are used as the input. In [11] and [5], it has been shown that the Mel log filter bank (FBANK) features outperform Mel frequency cepstral coefficients (MFCCs) because FBANK coefficients are less pre-processed, whereas MFCCs reduces the dimensionality of the input. Adding more hidden layers increases the number of trainable parameters of network, and it makes the model overfit easily [11]. Authors in [6] have demonstrated the effectiveness of pre-training in DNN, which can reduce over-fitting and required time for fine-tuning.

In the recent years, much work has focused on the training phase. Some researchers have investigated the effects of different types of network architecture [10], network depth, and number of hidden units [5, 11]. Others have focused on developing training algorithms, representing input by various features [9, 13, 14], overcoming issues related to training such as over-fitting [10, 15].

However, after training the neural network, during the test step, generating the optimal phoneme sequence is an important issue on which we focused in this work. In the recent research works, the Viterbi algorithm has been used on a hidden Markov model to obtain phoneme sequences as done in [10-12]. In the current work, our goal was to improve the neural network performance in the test step and to generate more accurate phoneme sequences. Consequently, three methods were investigated and compared. We present the empirical evaluations of these three methods using the Persian FarsDat corpus and compare their results with the GMM-HMM baseline method.

I. Post-processing method: We correct some errors caused by network after the detection of frame labels. This method leads to a better accuracy compared with the case where no post-processing is done. In this method, we investigate the feasibility and performance of correction of network results based on our knowledge about phonemes.

II. Viterbi algorithm on hidden Markov model: Viterbi algorithm is used to find the optimal phoneme sequence. Compared to the post-processing method, this method leads to a better performance.

III. Extended Viterbi algorithm on hidden semi-Markov model (HSMM): We use HSMM [16] to generate phoneme sequence in the test phase. As known, HSMM has been used successfully in many applications, and it is capable of defining a duration probability distribution for remaining in each state [16]. HMM can obtain phoneme sequence but it considers a geometric distribution for each state, which is different from the real phoneme duration distribution. We address this issue for DNN-based phoneme recognition using HSMM, which, as we know has not been studied previously. Our results show that by applying the extended Viterbi algorithm on HSMM, a better phoneme sequence can be obtained compared to HMM that has previously been used in the research work. This superiority is due to modeling of duration probability distribution for each phoneme by HSMM.

We consider a suitable topology for the HMM and HSMM models, where each phoneme is presented by a state, and this work makes using phoneme duration distribution easier in HSMM.

The rest of this paper is organized as what follows. Section 2 first explains phoneme recognition using deep neural networks and the two phases training and testing, and then presents three methods to find phoneme sequence during the test step including post-processing method, HMM and HSMM. Section 3 describes the corpus used in our experiments and the experimental setup. Section 4 shows the experimental results and analysis. Finally, conclusions and future work are presented in Section 5.

2. Phoneme recognition using deep neural networks

Phoneme recognition using deep neural network has two phases including training and testing. These steps are explained as follow, and are illustrated in Figure 1.

2.1. Training phase

The aim of this phase is to train each window of frames and its corresponding phoneme label by neural network. We use a window of n successive frames as the input of the neural network. The phoneme label of each window of frames is the phoneme label of the central frame that was determined based on the label of its time slot in the training corpus. As mentioned earlier, our goal was phoneme recognition for the Persian language with 32 phonemes.

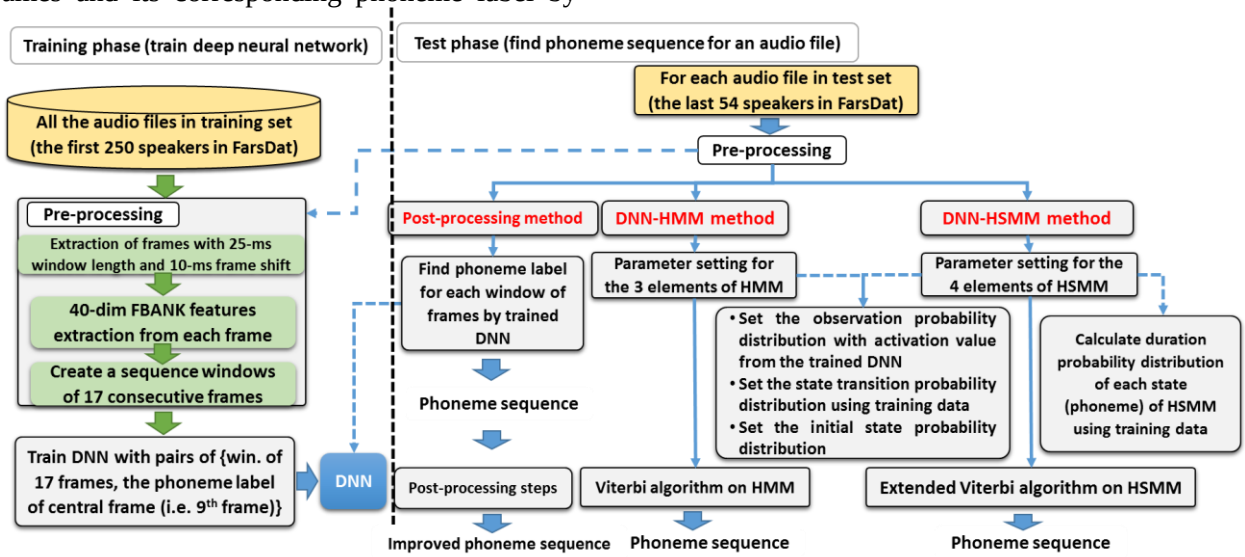

Figure 1. Block diagram for the proposed train and test procedures.

2.2. Testing phase

During the test phase, to find the phoneme sequence for an audio file, feature vectors are extracted from its speech frames. Then each window of frames is fed to neural network as the input, and the trained network generates a phoneme label for the central frame of this window. This label corresponds to one of 32 Persian phonemes. Phoneme labels for all frames of audio file are obtained. The frame level accuracy is calculated as the ratio of the number of correct recognized frames to the total number of frames. It is defined as:

$$\text{Frame level accuracy} = \frac{\#\text{Correct recognized frames}}{\#\text{Frames}} * 100 \quad (1)$$

In the next step, for each audio file, the labels of adjacent frames are merged, and labels of phoneme in each sequence are generated. In our experiments, the frame level accuracy was 84.44% but a low phoneme accuracy of 29.06% was achieved. These results show that despite true detection of labels of most of frames (i.e. 84.44%), phoneme sequence is not properly recognized due to the high number of insertion, which can greatly affect the results of phoneme recognition. In order to achieve a better phoneme recognition accuracy on the test speech files, we investigate three approaches including post-processing method, Viterbi algorithm on HMM, and extended Viterbi algorithm on HSMM; the first and third methods are proposed in this work. In the following, we explain these three approaches.

2.2.1. Generating phoneme sequence using post-processing method

We propose a post-processing method to generate a true phoneme sequence. In this method, each window of frames corresponding to the audio file is fed to neural network as the input and is labeled by neural network. Then the post-processing steps are performed on these results. In fact, by post-processing, we correct some errors of frame labels that significantly improve the phoneme recognition accuracy. Detection and correction of these errors are done using our knowledge about phonemes and syllable structures in the Persian language. In the following, we will introduce these errors and explain how to correct them. We show these errors in Figure 2.

Error type 1: During the experiments, sometimes it was observed that a phoneme label was assigned to some consecutive speech frames but a single

frame among them had a different label. Some examples of this kind of error are shown in cases (1), (2), and (3) in Figure 2. Duration of a phoneme is usually longer than the duration of one frame, and it is likely that the intermediate frame has received a wrong label. To correct this error, the label of intermediate frame is changed to the label of its adjacent frames. In the case (3) in Figure 2, we can change phoneme *b* to phoneme *a* or *c*. Both of these changes generate the phoneme sequence *'ac'*.

Error type 2: Sometimes errors like the cases (4), (5), (8), (9), and (10) in Figure 2 may occur. We explain how to correct these types of errors for case (4); the explanation for other cases is similar. In the case (4), *b* and *c* phonemes have only one frame duration, and they are probably wrong. In order to correct this error, one of the phoneme sequences *'abd'*, *'acd'* or *'ad'* can be selected, shown in Figure 2 with three cases A, B, and C, respectively.

To select among three cases A, B, and C, the case with the highest probability of phoneme sequence is selected. Here, the probability of a phoneme sequence is calculated by multiplication of phoneme bigram probabilities, which is calculated by the HTK toolkit [17]. For example, if the probability of *'abd'* phoneme sequence (i.e. the A case) is greater than the other sequences (e.g. B and C in Figure 2), then the A case (i.e. phoneme sequence *'abd'*) is selected. In the Figure 2, the value for $P(b|a)$ shows the probability of the 'ab' sequence.

Error type 3: We know that two vowels never occur successively but sometimes this kind of error occurs in the output phoneme sequence. Cases (6) and (11) in Figure 2 show this kind of error; here, both v_1 and v_2 are vowels. In case (6), another vowel has occurred between two vowels. In order to correct this error, two central frames that correspond to one vowel are changed to phoneme labels of adjacent frames. In case (11), we should change one of the vowels into another one.

The value for $act(v_1, fr)$ is an activation value for the frame fr and the vowel v_1 that is generated by the network. Starting from the first frame of first vowel to the frame where the second vowel ends (i.e. from i'th frame to l'th frame), the activation values of each vowel from its i'th frame to l'th frame are multiplied, and the greater one is selected to decide on the final vowel since its activation value is greater for all of these frames.

Error type 4: Sometimes there are two frames with non-silence frames among them since silence duration is usually long and the silence phoneme lasts some frames. It is impossible that two non-silence phonemes occur in this long period. Thus we change these non-silence phonemes to silence. This case is depicted in case (7) in Figure 2.

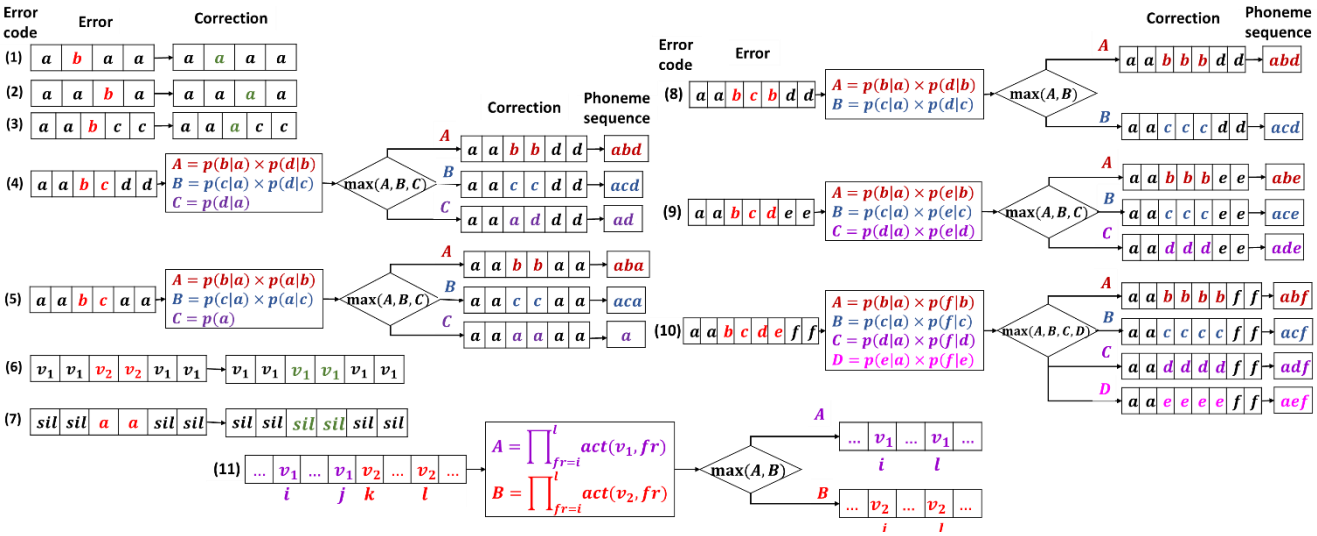

Figure 2. Steps of post-processing method: error codes, error types, and error correction methods.

This method is language-independent and can be used for any language. All of these cases are possible to occur in any language.

The symbol *a*, *b*, *c* or *d* does not show a specific phoneme, so these errors can happen for any phonemes. There may be doubt about cases (6) and (11). It is worthy to mention that the cases (6) and (11) may occur in English too since two vowels cannot follow each other in English.

2.2.2. Generating phoneme sequence using HMM

In this method, we use a HMM with 32 states, as shown in Figure 3, where each state represents one phoneme. Then Viterbi algorithm is used to generate phoneme sequence. Using the Viterbi algorithm, sequence of optimal states for sequences of windows of frames (observations) are obtained.

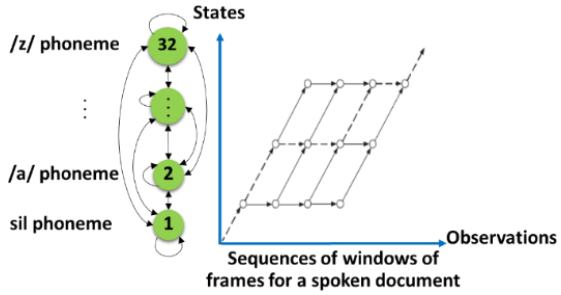

Figure 3. Performing Viterbi algorithm on HMM to find optimal phoneme sequence for a spoken document.

As shown in Figure 3, the HMM model is Ergodic. This model has the ability to produce any phoneme sequence. Therefore, each state that shows a phoneme should be connected to other states to generate any possible phoneme sequence. The three elements of HMM (B,A,π) are defined as follow:

Observation probability distribution (B): During the test step, for each window of frames as input to the network, each of the 32 output neurons (phoneme classes) has an activation value that shows the probability that the input belongs to phoneme classes. Therefore, probability distribution for window of frames (observation) is obtained for each state of HMM using these 32 values.

State transition probability distribution (A): Transition matrix between states has 32x32 dimensions. These transition probability values are obtained by calculation of the probability of transition between labels of two windows of consecutive frames in the training set. In other words, transition probability between states i and j is the number of transitions between two consecutive windows of frames from phoneme i to phoneme j divided by the number of transitions from phoneme i to other phonemes.

Initial state probability distribution (π): In order to define the initial state probability for HMM, we set the initial probability of the state that corresponds to silence phoneme to be one and other states to zero because the spoken files in the train and test sets begin with silence.

2.2.3. Generating phoneme sequence using HSMM

In this section, we first discuss a problem associated with HMM for DNN-based phoneme recognition task, and then explain how to solve this problem using HSMM.

Problem: An important weakness of HMM is that it cannot explicitly model the state duration. The state duration in HMM follows a geometric distribution [18] because the probability of observing d consecutive observations in state i conditioned on start from state i is:

$$P_i(d) = (1 - a_{ii})a_{ii}^{d-1} \quad (2)$$

where, a_{ii} is the self-transition probability of state i.

In phoneme recognition, our preliminary experiments show that the duration probability distribution of phonemes does not obey geometric distribution. Here, for example, duration probability distributions for vowel /a/ and consonant /b/ are depicted in Figure 4 and Figure 5 respectively.

Duration probability distribution of each phoneme is calculated based on the number of frames. In this experiment, the frame length is 25 ms and the frame shift is 10 ms. The horizontal axis indicates the number of frames elapsed for phoneme and the vertical axis indicates its duration probability.

Based on these figures, duration probability distribution of phonemes has a non-geometric distribution, whereas HMM considers a geometric distribution.

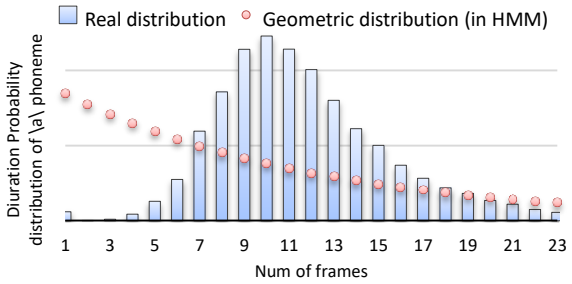

Figure 4. Duration probability distribution for vowel /a/ .

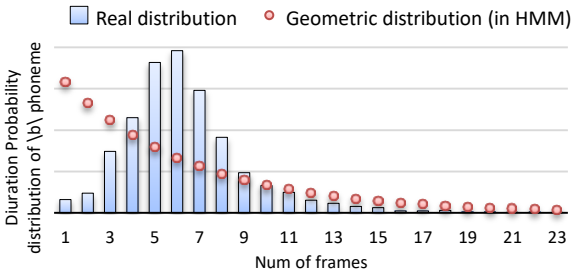

Figure 5. Duration probability distribution for consonant /b/.

In order to use the real duration probability of each phoneme for finding the phoneme sequence, as shown in the above figures, we use HSMM instead of HMM. A HSMM in an extension of HMM. HSMM is capable of explicitly defining a desired duration probability distribution to stay in each state. This model is known as semi-Markov since predicting the next state depends not only on the previous but also on the elapsed time in the states [18, 19]. It should be noted that in HMM used in Section 2.2.2, each state represents a phoneme; therefore, it would be easy to model the duration of each state, which is equivalent to the duration of each phoneme. HSMMs have different types depending on the considered assumptions. One kind of HSMM is the explicit-duration hidden Markov model, which is one of the simplest models among HSMM models due to considering some independency assumptions. Therefore, it is the most popular and applicable HSMM model [16, 20]. We explain this model in the following sub-section.

Explicit-duration hidden Markov model: The complete parameter set of this model is indicated by $\lambda = (\{a_{mn}\},\{\pi_m\},\{b_m(v_k)\},\{p_m(d)\})$. In addition to three parameters of HMM, this model has a probability distribution for the time remaining in states (i.e. $p_m(d)$) [16, 20].

Assume that the model has M states denoted with $s_1, s_2, ..., s_M$ and q_t shows the state at time *t*. o_t is the observable output at time t, where $t = 1, 2, ..., T$. The four elements of the model $\{a_{mn}, \pi_m, b_m(v_k), p_m(d)\}$ are defined as follow:

I. State transition probability from state s_m to state s_n is denoted by $a_{mn}(m, n = 1, 2, ..., M)$.

II. The initial state distribution is defined by $\{\pi_m\}$

III. The conditional probability distribution is denoted with $b_m(o_t) = \Pr[o_t | q_t = s_m]$ and by considering "conditional independency" of outputs given states, we have:

$$\Pr[o_a^b | s_m] = \prod_{t=a}^{b} b_m(o_t) \qquad (3)$$

where, $o_a^b = \{o_t; a \le t \le b\}$.

IV. The duration of a given state is a discrete random variable, and is defined by the probability $p_m(d)$, where $d \in \{1, 2, ...D\}$. The integer value of D is the possible maximum duration to remain in any state. In the case of our task, duration probability of each phoneme is calculated based on the number of elapsed frames for that phoneme, and is calculated using the training set.

Suppose that τ_t shows the required remaining time to stay in the current state q_t. If $(q_t, \tau_t) = (s_m, d)$, model stays in the current state s_m until, then goes to another state at time $t + d$ (where $d \ge 1$).

The Extended Viterbi algorithm is used to find the optimal state sequence in HSMM [16, 20]. We explain this algorithm in the following sub-section.

Extended Viterbi algorithm: In this algorithm, we need to define the quantity named $\delta_t(m, d)$, which is defined as:

$$\delta_t(m, d) = \max_{q_1^{t-1}} \Pr[q_1^{t-1}, o_1^t, (q_t, \tau_t) = (s_m, d) | \lambda] \qquad (4)$$

When the current state is s_m and residual time to remain in this state equals d, this quantity gives the best score (the highest probability) for state sequence and observation sequence at time t.

There are M possible paths to transit $(q_t, \tau_t) = (s_m, d)$, which are shown in **Error! Reference source not found.**.

Transition into state $(q_t, \tau_t) = (s_m, d)$ can occur within one of these two conditions:

1. From $(q_{t-1}, \tau_{t-1}) = (s_m, d+1)$: In this case, the semi-Markov chain was in state s_m at time $t-1$ (i.e. at the previous time) and should remain for $d+1$ time units in that state. Path m in **Error! Reference source not found.** shows this case.

2. From $(q_{t-1}, \tau_{t-1}) = (s_n, 1)$, where $n \ne m$: In this case, the semi-Markov chain was in state s_n $(n \ne m)$ at the previous time, and after remaining in state s_n for one time unit, model transits to another state s_m.

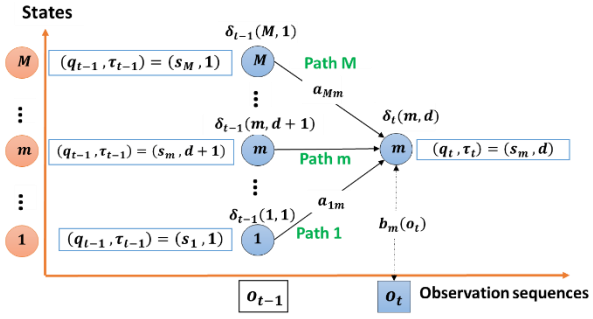

Figure 6. Illustration of performing Extended Viterbi algorithm on HSMM.

In **Error! Reference source not found.**, path 1 to M except m is a kind of these paths.

According to these M possible paths, we can induce the following recursion formula to find $\delta_t(m,d)$:

$$\delta_t(m,d) = \max\{\delta_{t-1}(m,d+1).b_m(o_t), \max_{n\in\{1,2,...M\}|n\neq m}[\delta_{t-1}(n,1)a_{nm}].b_m(o_t).p_m(d)\} \quad (5)$$

In order to record the state sequence and the duration of each state of optimal path, the variable $\psi_t(m,d)$ is used. Finally, the procedure of finding optimal path of state sequence is stated as follows:

I. Initialization: Initial condition for each state s_m where $m\in\{1,2,...M\}$ and duration $d\in\{1,2,...D\}$ is defined as:

$$\delta_1(m,d) = \pi_m.b_m(o_1).p_m(d) \quad (6)$$

$$\psi_1(m,d) = (0) \quad (7)$$

II. Recursion: In this step, the values for $\delta_t(m,d)$ and $\psi_t(m,d)$ for each time $2\le t\le T$ and state s_m where $m\in\{1,2,...M\}$ and duration $d\in\{1,2,...D\}$ are calculated by:

$$v = \max_{n\in\{1,2,...M\}|n\neq m}[\delta_{t-1}(n,1)a_{nm}].b_m(o_t).p_m(d) \quad (8)$$

$$i = \arg\max_{n\in\{1,2,...M\}|n\neq m}[\delta_{t-1}(n,1).a_{nm}] \quad (9)$$

$$\delta_t(m,d) = \max\{\delta_{t-1}(m,d+1).b_m(o_t), v\} \quad (10)$$

$$\psi_t(m,d) = \begin{cases} i, & \delta_{t-1}(m,d+1).b_m(o_t) < v \\ m, & \text{else} \end{cases} \quad (11)$$

III. Termination:

$$(q_T^*,d_T^*) = \arg\max_{\substack{m\in\{1,2,...M\}\\ d\in\{1,2,...D\}}} \delta_T(m,d) \quad (12)$$

IV. Backtracking (finding optimal path):

$$(q_t^*,d_t^*) = \psi_{t+1}(q_{t+1}^*,d_{t+1}^*),\ t = T\text{-}1,\ T\text{-}2,\ ...,1 \quad (13)$$

3. Experimental setup

In this section, we explain the detail of simulation for the neural network, and then introduce the dataset used for the experiments. After that, we present evaluation of a phoneme recognition system.

3.1. Detail of simulation

We will explain the detail of training neural network including its input and output, its architecture, and the training procedure:

Input and output of neural network: In our experiments, the input of the neural network comprises a window of 17 consecutive frames. In [11], the best choice for the number of consecutive frames is 11, 17 or 27 frames. A window of 17 frames is neither too short nor too long, and it contains 170 ms of speech signal and can provide enough information about the context around a phoneme. We used 25 ms frame windows, which spanned every 10 ms. We extracted 40 FBANK features from each frame due to the better performance of FBANK compared to MFCC for the phoneme recognition task [5, 6, 11]. The frame shift between two consecutive windows equals one frame. Thus the number of neurons in the input layer is the number of frames in a window multiplied by the dimension of the feature vector. The feature vectors at the input of the network were normalized to have zero mean and unit variance using the following equation:

$$\hat{x}_m = \frac{x_m - \mu_m}{\sigma_m} \quad (14)$$

where, μ_m and σ_m are, respectively, the mean and standard deviation of the feature x_m estimated using the training set.

There are 31 phonemes in the Persian language including two phonemes for two different sounds of /k/ and /g/. Silence is considered as the 32th phoneme. Therefore, there will be 32 neurons in the network output layer corresponding to 32 Persian phonemes [21].

Neural network architecture: In many recent research works [11], it has been shown that the deep neural networks outperform the shallow ones. Therefore, in this research work, a deep neural network with the parameters mentioned in Table 1 was configured and used in our experiments. The structure of this network is depicted in Figure 7.

Training neural network: In order to train the phoneme recognition system, we used the DeeBNet MATLAB toolbox [22] which provides tools for training Deep Belief Networks (DBNs). The network was first pre-trained, and was then fine-tuned by the back-propagation technique.

Table 1. Configuration of deep neural network.

Feature type	FBANK
Number of frames in each window	17
Number of hidden layers	4
Number of hidden neurons in each layer	1024
Frame feature vector dimension	40

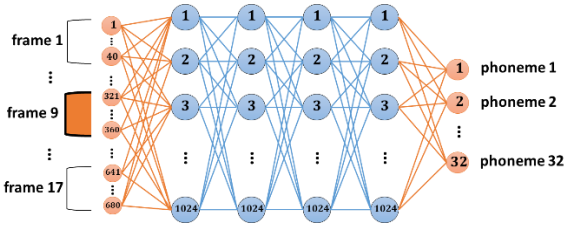

Figure 7. Neural network architecture for phoneme recognition task. The network input comprises a window of 17 consecutive frames including 40 dimensional FBANK features. The output label is the phoneme label of central frame (i.e. 9th frame).

3.2. FarsDat Corpus

We used FarsDat [23] to train a phoneme recognition system for the Persian language, which contains the utterances of 304 female and male speakers from 10 dialect regions in Iran. FarsDat is tagged in phoneme level, and it is similar to TIMIT used for English phoneme recognition. Each speaker uttered 20 sentences in two sessions. The utterances of first 250 speakers were used as the training set and utterances of the remaining 54 speakers were used as the test set. Our method is speaker-independent because the speakers in the training set are different from those in the test set. In FarsDat, each speaker utters 18 sentences extracted randomly from a set of 384 phonetically balanced sentences, so there may be similar sentences in the training and test sets. There are also two sentences uttered by all speakers. Therefore, some context dependency may exist in our experiments.

3.3. Evaluation of a phoneme recognizer

In order to evaluate a phoneme recognizer, the recognized and reference phoneme label sequence should be compared. The tool HResults in HTK [17] compares two strings by matching using dynamic programming. Considering the number of substitution errors, deletion errors and insertion errors shown as S, D, and I, respectively, the phoneme recognition accuracy is defined as:

$$\text{Accuracy} = \frac{\text{N-S-D-I}}{\text{N}} * 100\% \qquad (15)$$

where, N is the total number of labels in the reference label [17].

4. Results and discussion

In this section, we first present the results of the GMM-HMM baseline method. Then we investigate the results of three methods including post-processing, HMM and HSMM in the DNN-based phoneme recognizer.

4.1. Results of GMM-HMM baseline

We used 25 ms frame windows with 10-ms frame shifts. We extracted 12th-order Mel frequency cepstral coefficients (MFCCs) and energy along with their first and second temporal derivatives from each frame. Each phoneme was modelled by a 3-state left to right HMM with 2, 4, 8, and 16 mixtures of Gaussian per state.

We used the bigram language model (LM) to obtain the best result of the baseline GMM-HMM, and compared it with our proposed method. Phoneme language models estimate the probability of a phoneme sequence, and they are built using a training text. A phoneme n-gram LM is used to predict each phoneme in the sequence given its n-1 predecessors [17].

Using the bigram language model in Viterbi decoder is a very common method that is widely used in different research works like [5, 11] for the English and [12, 21] Persian languages. Besides, Bigram LM is commonly used in the research works due to its facility and simplicity compared with trigram in decoding time for real time applications.

The results for different number of mixtures of Gaussian per state are shown in Figure 8, and the effect of bigram LM is also investigated.

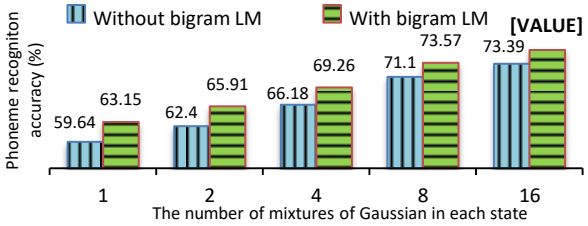

Figure 8. Phoneme recognition accuracy using GMM-HMM baseline method.

It can be seen that increasing the number of mixtures of Gaussian improves the performance. However, this increase can improve the accuracy to some extent. In [21], 16 mixtures of Gaussian per state were a proper value; therefore, it was selected for this work. Finally, the results demonstrate that the best performance is 75.83% using the GMM-HMM method.

4.2. Results of phoneme recognition using DNN

In this section, we aim to compare three methods for generating phoneme sequence including post-processing, HMM, and HSMM. Firstly, we show the result of each step of the post-processing method. Then we compare the results of the three methods and the GMM-HMM baseline method.

4.2.1. Results of generating phoneme sequence using post-processing method

The results of applying post-processing method on the output of neural network are shown in Table 2. Each error code corresponds to one of the codes of the post-processing method in Figure 2. Here, code (0) shows the results before applying the post-processing steps. We see from the results that before applying post-processing steps (i.e. in step (0)), all kinds of errors, especially insertion error, are very high, and after applying correction steps, many insertion errors decrease; therefore, phoneme recognition accuracy increases, because in these cases, sometimes only one inserted phoneme in many times can increase the number of insertion error.

All these 11 correction steps helped to decrease insertion error, and the best phoneme recognition accuracy is achieved in the last step. The frame level accuracy also improves until step 10 but it suddenly decreases in step 11, whereas phoneme recognition accuracy increases in this step. The reason is that in order to correct the error (11), which was discussed in the Section 2.2.1, one of the vowels v_1 or v_2 was selected, and all the frames were replaced by the selected vowel from the frame that the first vowel started, to the frame that second vowel ended. However, if the selected vowel has been recognized wrong, it causes the low error in the phoneme sequence because only one phoneme is recognized to be false, whereas many frames are damaged in the frame level.

Table 2. Results of applying each step of post-processing method to generate phoneme sequence.

Error code	Frame level accuracy (%)	Phoneme recognition accuracy (%)	#Insertions	#Deletions	#Substitutions	#Recognized phonemes
0	84.44	29.06	20590	505	2366	30202
1	84.66	42.87	15793	656	2445	29972
2	84.71	46.50	14530	697	2467	29909
3	84.86	57.73	10299	947	2734	29392
4	84.91	60.83	9142	1013	2801	29259
5	84.92	61.17	9007	1032	2803	29238
6	84.98	62.88	8406	1060	2812	29201
7	84.99	62.97	8364	1078	2806	29189
8	84.99	63.38	8202	1094	2815	29164
9	84.99	64.01	7973	1099	2831	29143
10	84.99	64.09	7942	1099	2835	29139
11	83.73	**68.03**	5473	1424	3675	27974

4.2.2. Comparison of results of generating phoneme sequence using post-processing, HMM and HSMM methods

The results of the three methods to generate phoneme sequence in DNN-based phoneme recognition system including post-processing, HMM and HSMM are presented in figure 9

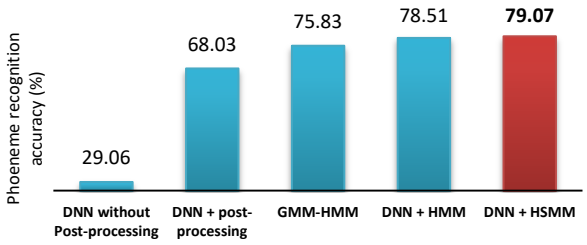

Figure 9. Results of phoneme recognition accuracy using three methods to generate phoneme sequence (post-processing, HMM and HSMM) in DNN-based phoneme recognition, and comparing their results with GMM-HMM baseline method.

The results of the GMM-HMM baseline method are also shown in the figure for a comparison. In this figure, "DNN without Post-processing" shows the performance before applying one of the three methods (i.e. post-processing, HMM and HSMM).

We observe from **Error! Reference source not found.** that before applying the three methods, the accuracy is very low, and it increases by using all the three methods. The post-processing method highly increases the accuracy compared to before its application, so it is possible to improve the results obtained from neural network. Applying Viterbi algorithm on the output activation of the neural network can find the optimal phoneme sequence but the post-processing method causes the local optimization in consecutive phonemes, whereas Viterbi algorithm finds the optimal phoneme sequence over all frames. The post-processing method can detect some wrong outputs

of network and correct them, while Viterbi algorithm can automatically correct many errors. Using both HMM and HSMM within neural network could result in significant improvement compared to the GMM-HMM baseline method. As expected, HSMM outperforms the other methods, especially HMM. This is consistent with our expectation in Section 2.2.3 that HSMM is better than HMM because we used real phoneme duration distribution for decoding in HSMM, while HMM considers a pre-defined geometric distribution for duration of phonemes, which is different from real distribution. Next, for a better comparison between HMM and HSMM, we show the results of the number of insertions, deletions, and substitutions for these methods in Figure 10.
We observe that using HSMM helps to decrease in the insertion and deletion errors, whereas for substitution errors, there is less increasing. Thus using information on phonemes duration for decoding in HSMM can prevent the extra insertion or deletion of phonemes, and it can result in a better phoneme recognition accuracy.

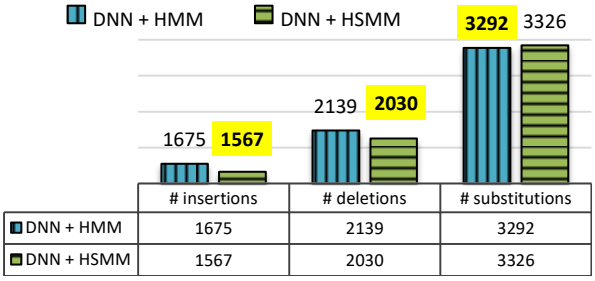

Figure 10. Number of insertions, deletions, and substitutions for HMM and HSMM methods.

It is worthy to note that by considering deep architecture for neural network, better recognition performance can be achieved using rich features, the suitable number of input frames, etc., and these improvements are related to the training phase. However, in order to increase the accuracy in the test phase, we should improve generating phoneme sequence, so we used HSMM. The improvement using HSMM is independent from the methods used in the training phase, and in the case of availability of the best trained network, the optimal phoneme sequence can be achieved using HSMM, while it has a better efficacy than both the HMM and GMM-HMM baseline methods.

5. Conclusion and future work

In this work, we presented our work on phoneme recognition using deep neural network, which has two phases including training and testing. Most of the previous research works have focused on improving the training phase and they have used HMM to find the phoneme sequence during the test phase. However, HMM uses a geometric distribution for phoneme duration, whereas real phoneme duration distribution does not follow it. To address this problem, in this work we used HSMM with a suitable topology, where each state equals to one phoneme. Our experimental results on the FarsDat corpus showed that using HSMM can improve phoneme recognition accuracy compared to the HSMM and GMM-HMM baseline methods. This gain is from the use of duration probability distribution for each phoneme, estimated from the training set, in HSMM. We also proposed a post-processing method, which can correct some errors from the neural network, based on our knowledge about phonemes. Although our experiments showed that the accuracy of the post-processing method was lower than the GMM-HMM method, the correction steps of this method could improve phoneme sequence compared to before its application. Thus in our future work, we plan to explore more rules for this method and also combine rules of post-processing with the HSMM method to generate a more accurate phoneme sequence in the test phase.

References

[1] Chen, N. F., Ma, B. & Li, H. (2013). Minimal-resource phonetic language models to summarize untranscribed speech. IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP), Vancouver, BC, Canada.

[2] James, D. A. & Young, S. J. (1994). A fast lattice-based approach to vocabulary independent wordspotting. ICASSP, Adelaide, SA, Australia.

[3] Hazen, T. J., Richardson, F. & Margolis, A. (2007). Topic identification from audio recordings using word and phone recognition lattices. Automatic Speech Recognition and Understanding (ASRU), Kyoto, Japan.

[4] Lee, L.-s., et al. (2015). Spoken Content Retrieval—Beyond Cascading Speech Recognition with Text Retrieval. IEEE Transactions on Audio, Speech, and Language Processing**,** vol. 23, no. 9, pp. 1389-1420.

[5] Jaitly, N. PhD thesis. (2014). Exploring Deep Learning Methods for discovering features in speech signals. University of Toronto.

[6] Hinton, G., et al. (2012). Deep neural networks for acoustic modeling in speech recognition: The shared views of four research groups. IEEE Signal Processing Magazine**,** vol. 29, no. 6, pp. 82-97.

[7] Dahl, G. E., et al. (2012). Context-dependent pre-trained deep neural networks for large-vocabulary speech recognition. IEEE Transactions on Audio,

Speech, and Language Processing**,** vol. 20, no. 1, pp. 30-42.

[8] Darabian, D., Marvi, H. & Sharif Noughabi, M. (2015). Improving the performance of MFCC for Persian robust speech recognition. Journal of AI and Data Mining**,** vol. 3, no. 2, pp. 149-156.

[9] Mohamed, A.-r., Hinton, G. & Penn, G. (2012). Understanding how deep belief networks perform acoustic modelling. ICASSP, Kyoto, Japan.

[10] Mohamed, A.-r., Dahl, G. & Hinton, G. (2009). Deep belief networks for phone recognition. NIPS workshop on deep learning for speech recognition.

[11] Mohamed, A.-r., Dahl, G. E. & Hinton, G. (2012). Acoustic modeling using deep belief networks. IEEE Transactions on Audio, Speech, and Language Processing**,** vol. 20, no. 1, pp. 14-22.

[12] Ansari, Z. & Salehi, S. A. S. (2014). Proposing two speaker adaptaion methods for deep neural network based speech recognition systems. 7th International Symposium on Telecommunications (IST), Tehran, Iran.

[13] Mohamed, A. R., et al. (2011). Deep belief networks using discriminative features for phone recognition. ICASSP, Prague, Czech Republic.

[14] Dahl, G., Mohamed, A.-r. & Hinton, G. E. (2010). Phone recognition with the mean-covariance restricted Boltzmann machine. the 23rd International Conference on Neural Information Processing Systems (NIPS), Vancouver, British Columbia, Canada.

[15] Srivastava, N., et al. (2014). Dropout: a simple way to prevent neural networks from overfitting. Journal of Machine Learning Research**,** vol. 15, no. 1, pp. 1929-1958.

[16] Yu, S.-Z. (2010). Hidden semi-Markov models. Artificial Intelligence**,** vol. 174, no. 2, pp. 215-243.

[17] Yong, S., et al., (2009). The HTK book. Cambridge University Press.

[18] Rabiner, L. R. (1990). A tutorial on hidden Markov models and selected applications in speech recognition. Readings in Speech Recognition**,** vol. 77, no. 2, pp. 267-296.

[19] Murphy, K. P. PhD thesis. (2002). Dynamic bayesian networks: representation, inference and learning. University of California, Berkeley.

[20] Yu, S.-Z. & Kobayashi, H. (2003). An efficient forward-backward algorithm for an explicit-duration hidden Markov model. IEEE Signal Processing Letters**,** vol. 10, no. 1, pp. 11-14.

[21] Homayounpour, M. M., et al. (2008). A Very low bit rate Phonetic Vocoder for Farsi Language. Signal and Data Processing**,** vol. 8, no. 2, pp. 3-26 (in Farsi).

[22] Keyvanrad, M. A. & Homayounpour, M. M. (2014). A brief survey on deep belief networks and introducing a new object oriented MATLAB toolbox (DeeBNet). arXiv:1408.3264, 2014.

[23] Bijankhan, M., Sheikhzadegan, J. & Roohani, M. R. (1994). FARSDAT-The speech database of Farsi spoken language. 5th Australian conference on speech science and technology, Canberra.

نشریه هوش مصنوعی و داده کاوی

بهبود بازشناسی دنباله واج با استفاده از اطلاعات زمانی واج در DNN-HSMM

مریم اسداله زاده کرمانشاهی و محمد مهدی همایونپور*

دانشکده مهندسی کامپیوتر و فناوری اطلاعات، دانشگاه صنعتی امیرکبیر، تهران، ایران.

ارسال ۲۰۱۷/۰۸/۱۷؛ بازنگری ۲۰۱۸/۰۲/۲۳؛ پذیرش ۲۰۱۸/۰۵/۱۹

چکیده:

بهبود بازشناسی واج به دلیل کاربردهای متنوع در پردازش گفتار توجه بسیاری از پژوهشگران را به خود جلب کرده است. تحقیقات اخیر بیانگر این است که استفاده از شبکه‌های عصبی عمیق (DNN) در بازشناسی گفتار کارایی را به طرز چشمگیری افزایش می‌دهد. سیستم‌های بازشناسی واج مبتنی بر DNN، شامل دو مرحله‌ی آموزش و آزمایش هستند. در این مقاله، روی بهبود مرحله‌ی آزمایش که به ایجاد دنباله واجی مربوط است تمرکز شده است. این درحالیست که اکثر تحقیقات در جهت بهبود مرحله‌ی آموزش شامل الگوریتم‌های آموزش، نوع شبکه، معماری شبکه، ویژگی‌ها و ... گام برداشته‌اند. در تحقیقات گذشته برای تولید دنباله واجی از اجرای الگوریتم ویتربی روی مدل مخفی مارکف (HMM) استفاده شده است. محدودیتی که در استفاده از این مدل وجود دارد درنظرگرفتن توزیع هندسی برای حالت‌ها است. در تحقیق حاضر برای حل این مسئله، با کمک مدل مخفی شبه مارکف (HSMM) از توزیع احتمال زمانی واج‌ها استفاده می‌شود و به منظور سهولت استفاده از اطلاعات زمانی واج‌ها، هر واج با استفاده از یک حالت نمایش داده می‌شود. همچنین روشی به نام روش پس‌پردازش ارایه شده است که با کمک دانش واجی سعی در اصلاح دنباله واجی حاصل از DNN دارد. نتایج آزمایشات روی پیکره فارس‌دات نشان می‌دهد که استفاده از الگوریتم ویتربی گسترش‌یافته روی HSMM در مقایسه با روش مرسوم مدل مخلوط گاوسی- مدل مخفی مارکف (GMM-HMM) و روش اجرای ویتربی روی HMM به ترتیب موجب بهبود کارایی به میزان ۲/۶۸٪ و ۵۶/..٪ شده است. روش پس‌پردازش نیز نسبت به قبل از اعمال آن نتایج چشمگیری را نشان می‌دهد.

کلمات کلیدی: بازشناسی واج، شبکه عصبی عمیق، مدل مخفی مارکف، مدل مخفی شبه مارکف، الگوریتم ویتربی گسترش یافته، دیرش واج، زبان فارسی.