International Journal of Computer Applications (0975 – 8887)
Volume 5– No.8, August 2010

Search Key Identification in a Spoken Query using Isolated Keyword Recognition

Utpal Bhattacharjee
Department of Computer Science and Engineering,
Rajiv Gandhi University, Rono Hills, Doimukh,
Arunachal Pradesh, India, Pin-791 112,

ABSTRACT

This article presents a novel technique for the recognition of isolated keywords from spoken search queries. Recognition of the isolated keywords from spoken search queries may be considered as the first step towards the development of a speech-operated keyword-based searching technique. A database of 300 spoken search queries from Assamese language, a major Indian language mostly spoken by the people of north east India, has been created. The system developed during the study has been tested and evaluated with the above mentioned database. In the present study, Mel Frequency Cepstral Coefficient (MFCC) has been used as the feature vector and Multilayer Perceptron (MLP) to identify the phoneme boundaries as well as for recognition of the phonemes. Viterbi search technique has been used to identify the keywords from the sequence of phonemes generated by the phoneme recognizer. A recognition accuracy of 74.67% has been achieved in the present study.

Keywords

Query Identification, Phoneme Segmentation, Multilayer Perceptron, Viterbi Search

1. INTRODUCTION

Speech-based search queries are made on the basis of some specific keywords. Identification of the keywords in a search query is helpful in identifying the search key. There are two approaches for isolated word recognition – recognition of the word as a whole and recognition of the phonemes associated with the isolated word and then recognizes the word associated with the sequence of phonemes. In Shino-Tibetan family of languages where numbers of words are relatively less, the first approach is a suitable one. However, for Indo-Aryan family of languages like Assamese language, it is not feasible to use separate model for each word due to the large number of possible words. Since the number of phonemes is very less compared to the number of words, the second approach is most suitable for such languages. However, a major difficulty is such model is the identification of the phoneme boundary. A variety of methods have been proposed to accomplish this phoneme segmentation [8, 13, 15]. Most of the methods rely heavily on a series of acoustic phonetic rules. Since the rules are difficult to generalized, their performance degrades in real world applications. In order to overcome these problems a neural network based approach has been proposed in this paper. The neural network based approach being a non-parametric method, has advantage over rule based approaches and produces robust performance under unexpected environmental condition. Many neural network based attempts have been made for phoneme segmentation and some encouraging results have been reported [3, 6, 10]. In this paper a MLP-based segmentation method has been utilized.

Mel-frequency cepstral coefficients (MFCC) are extensively used and have proven to be successful for Automatic Speech and Speaker Recognition system. In the present work MFCC has been used as feature vector. To avoid excessive computational load for feature extraction, the same feature set has been used for both segmentation and recognition purpose.

The use of multi-layer perceptron as speech recognizer has been encouraged by many workers [1, 9, 11, 12] during the last few decades. The most obvious way to use multi-layer perceptron for speech recognition is to present all acoustic vectors of a speech unit (phoneme or word) at once at the input and detect the most probable speech unit at the output by determining the output neuron with highest activation. The problem associated with this approach is that a huge number of input units have to be used, which implies evenly larger number of parameters required to be determined by learning and consequently the necessity to dispose of a large database. To reduce the volume of input data Kohonen self-organized map [7] (SOM) have been used in this study.

The paper is organized into the follow sections: Section 2 presents the brief description of the suggested architecture for the automatic keyword recognizer. Section 3 is devoted to the state of the art and mathematical background of MFCC parameterization, self-organized map, multilayer perceptron and Viterbi search algorithm used in the present study. The experiment and performance evaluation of the system is presented in Section 4. Section 5 concludes and presents perspective of this study. The last section lists the main references which have been used in this work.

2. ARCHITECTURE OF ISOLATED KEYWORD RECOGNIZER

Fig. 1 represents the main processing elements of the Isolated Keyword Recognizer. The first step for automatic speech recognition is to represent the speech signal in terms of feature vector. A feature vector is usually computed from a window of speech signal in every short time interval. An utterance is represented as a sequence of these feature vectors. In the present study, the speech signal is blocked into frames of 30 ms at an interval of 10 ms from which Mel-frequency cepstral coefficient has been calculated. The time derivatives of the MFCC are also appended to capture the dynamics of speech. In the present study MFCC coefficients along with its first order derivatives have been considered as the feature vector.

The feature vector extracted from the speech signal has been used for two purposes – phoneme segmentation and phoneme recognition. To use the feature vector for phoneme segmentation, a new feature set has been derived from the original MFCC feature set. This feature set is based on the difference between the feature vectors extracted from two consecutive frames. In the present study, we call it Differential MFCC (DMFCC). Differential Feature Extractor block is responsible for generating this set of feature vector. The DMFCC has been used as input to the Phoneme Segmenter block. A multilayer perceptron with one output unit has been used for this purpose. This block will return the frame number of the frames which contain a phoneme boundary.

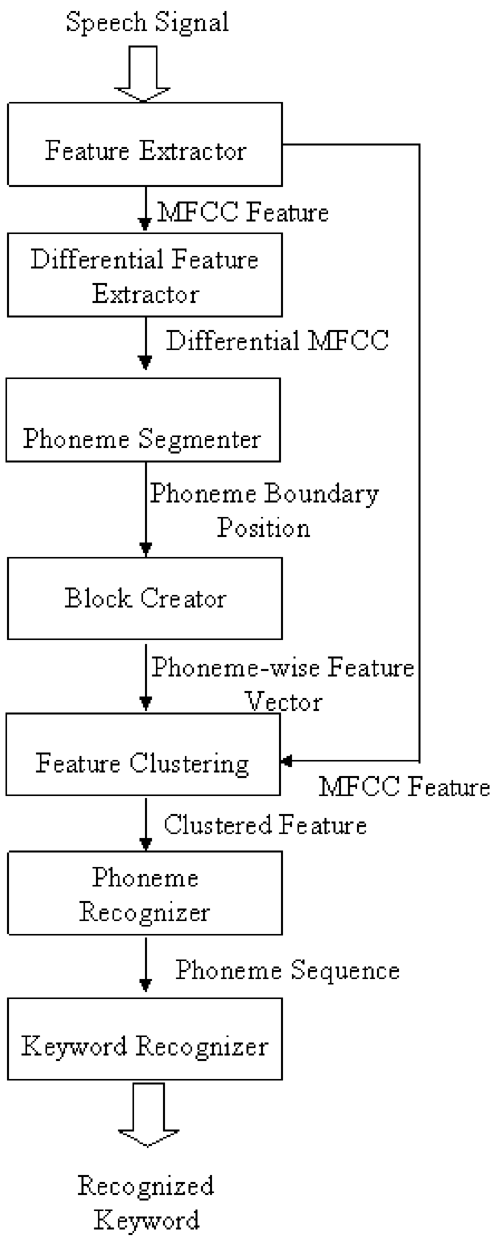

Fig. 1: Block diagram of the Isolated Keyword Recognizer

The next block is responsible for blocking the frames that belongs to a particular phoneme. The frames associated with each phoneme are clustered into six clusters using self-organized map (SOM). The output of the SOM is then considered as input to the Phoneme Recognizer block. The block is responsible for recognizing the phoneme. A multilayer perceptron has been used for this purpose. The output of this block is a sequence of phonemes associated with the uttered phrase. Viterbi search technique has been used to recognize the keywords associated with the sequence of phonemes.

3. MATHEMATICAL BACKGROUND AND ALGORITHMS USED

3.1 Mel-Frequency Cepstral Coefficient

The speech signal is divided into frames where discrete Fourier transform (DFT) has been computed for each frame. For the discrete time signal *x(n)* with length *N*, the DFT is given by

$$X(k) = \sum_{n=0}^{N-1} w(n)x(n)\exp(-j2\pi kn/N) \qquad (1)$$

for k = 0, 1, ….., N-1, where k corresponds to the frequency $f(k) = kf_k / N$, f_k is the sampling frequency in Hertz and *w(n)* is a time-window. In the present study Hamming windows defined by $w(n) = 0.54 - 0.46\cos(2\pi n/N)$ has been used because of its computational simplicity.

The magnitude spectrum $|X(k)|$ is now scaled in both frequency and magnitude. First the frequency is scaled logarithmically using the Mel filter bank *H(k,m)* and then the logarithm is taken giving

$$X'(m) = \ln\left(\sum_{k=0}^{N-1} |X(k)| . H(k,m)\right) \qquad (2)$$

for m = 1,2,……, M, where M is the number of filter banks and M << N. The Mel filter bank is a collection of triangular filters defined by the center frequencies $f_c(m)$, written as

$$H(k,m) = \begin{cases} 0, f(k) < f_c(m-1) \\ \dfrac{f(k) - f_c(m-1)}{f_c(m) - f_c(m-1)}, \\ \qquad f_c(m-1) \le f(k) < f_c(m) \\ \dfrac{f(k) - f_c(m+1)}{f_c(m) - f_c(m+1)}, \\ \qquad f_c(m) \le f(k) < f_c(m+1) \\ 0, f(k) \ge f_c(m+1) \end{cases} \qquad (3)$$

The center frequencies of the filter bank are computed by approximating the Mel scale with

$$\phi = 2595\log_{10}\left(\frac{f}{700} + 1\right) \qquad (4)$$

which is a common approximation. That equation is non-linear for all frequencies. Then a fixed frequency resolution in the Mel scale is computed, corresponding to a logarithmic scaling of the repetition frequency, using $\Delta\phi = (\phi_{\max} - \phi_{\min})/(M+1)$ where $\phi_{\max}$ is the highest frequency of the filter bank on the Mel scale, computed from $f_{\max}$ using equation (4), $\phi_{\min}$ is the lowest frequency in Mel scale, having a corresponding $f_{\min}$ and M is the number of filter banks. The values considered for the parameters in the present study are: $f_{\max}$=11.025 KHz, $f_{\min}$=0 Hz and M=30. The center frequencies on the Mel scale are given by $\phi_c(m) = m.\Delta\phi$ for m = 1, 2, 3, ….., M. To obtain the center frequencies in Hertz, inverse of the equation (4) is applied, which is given by

$$f_c(m) = 700\left(10^{\phi_c(m)/2595} - 1\right) \tag{5}$$

Equation (5) is inserted into equation (3) to give the Mel filter bank. Finally, the MFCCs are obtained by computing the discrete cosine transform of $X'(m)$ using

$$c(l) = \sum_{m=1}^{M} X'(m)\cos\left(l\frac{\pi}{M}\left(m - \frac{1}{2}\right)\right) \tag{6}$$

for l = 1, 2, 3, ….., M where $c(l)$ is the l^{th} MFCC.

The time derivative is approximated by a linear regression coefficient over a finite window, which is defined as

$$\Delta\ c_t(l) = \left[\sum_{K=2}^{2} k\ c_{t-k}(m)\right].G,\ 1 \le l \le M \tag{7}$$

where $c_t(l)$ is the l^{th} cepstral coefficient at time t and G is a constant used to make the variances of the derivative terms equal to those with the original cepstral coefficients.

3.2 Self Organized Map

Self Organized Map (SOM) were initially introduced with the purpose of producing a special mapping from a high dimensional input space to a very low dimensional output space while conserving the topological information of the input space. SOM is a neural network trained by following a non-supervised algorithm. The neural network is made up of two layers of neurons. The input (sensory) layer just distributes the inputs to the output layer. The output layer has M^p neurons, arranged on a p-dimensional lattice or map. Neurons on the output layer are characterized by:

1. A map coordinate $k=(k_1, k_2, k_3, \dots, k_p)$ with $1 \le k_1, k_2, k_3, \dots, k_p \le M$, that locates the neurons on the map
2. A synaptic weight vector $w_k = (w_k^1, w_k^2, \dots, w_k^m)$, where m is the neurons input dimension.

At a time n, an m-dimensional input vector $X= [x_1, x_2, x_3, \dots, x_m]^T$ is presented to the input of the network. The following sequence of operations taken place:

3.2.1 Competitive Process

The synaptic weight vector of each neuron in the network has the same dimension as the input space. The synaptic weight vector of neuron j is denoted by:

$$w_j = [w_{j1}, w_{j2}, w_{j3}, \dots, w_{jm}]^T,\ j = 1,2,\dots,l \tag{8}$$

where l is the total number of neurons in the network. To find the best match of the input vector X with the synaptic weight vectors W_j, the inner product W_j^TX for j=1,2,3, …. , l have been computed and neuron with highest inner product W_j^TX has been selected as the best match. Maximization of the inner product is taken as mathematically equivalent to the minimization of the Euclidean distance between the vectors X and W_j. Thus, if the index $i(x)$ is used to identify the neuron that best matches the input vector X, the following conditions is applied to identify $i(x)$:

$$i(x) = \arg\min_j \left\| x - w_j \right\|, j = 1,2,3.......l \tag{9}$$

which sums up the essence of the competition process among the neurons. The particular neuron i, that satisfies this condition is called the best match neuron for a typical input vector X.

3.2.2 Cooperative Process

The winning neuron locates the center of a topological neighbourhood. The probability that a neuron, that is firing tends to excite the neurons in its immediate neighbourhood is more than those farther away from it. This observation leads to make the topological neighbourhood around the winning neuron i decaying smoothly with lateral distance. To be specific, let $h_{j,i}$ denote the topological neighbourhood centered on winning neuron i, and encompassing a set of excited (cooperative) neurons, a typical one denoted by j. Then the topological neighbourhood $h_{j,i}$ is a unimodel function of lateral distance $d_{j,i}$ such that it satisfies two distinct requirements:

1. The topological neighbourhood $h_{j,i}$ is symmetric about the maximum point defined by $d_{j,i}$=0. In other words, it attains its maximum value at the winning neuron i for which the distance $d_{j,i}$ is zero.
2. The amplitude of the topological neighbourhood $h_{j,i}$ decreases monotonically with increasing lateral distance $d_{j,i}$, decaying to zero for $d_{j,i} \to \infty$, which is a necessary condition for convergence.

A typical choice of $h_{j,i}$ is a Gaussian Function as described by:

$$h_{j,i(x)} = \exp\left(-\frac{d_{j,i}^2}{2\sigma^2}\right) \tag{10}$$

where parameter σ is the "effective width" of the topological neighbourhood. It measures the degree to which excited neurons in the vicinity of the winning neuron participates in the learning process.

For cooperation among the neighbourhood neurons to be hold, it is necessary that topological neighbourhood $h_{j,i}$ must be dependent on lateral distance $d_{j,i}$ between winning neuron i and excited neuron j in the output space rather than as on some distance measure on the original input space. This has been achieved through the Equation (10). In the case of one dimensional lattice, $d_{j,i}$ is an integer equal to $| j – i |$. On the other hand, in the case of two dimensional lattice, it is defined by

$$d_{j,i}^2 = \| r_j - r_i \|^2 \tag{11}$$

where the discrete vector r_j defines the position of the excited neuron j and r_i defines the discrete position of the winning neuron, both measuring in discrete output space.

Another unique feature of the SOM algorithm is that the size of the topological neighbourhood shrinks with time. This requirement is satisfied by making the width σ of the topological neighbourhood function $h_{j,i}$ decreases with time. A popular choice for the dependence of σ on discrete time n is the exponential delay described by equation given below:

$$\sigma(n) = \sigma_0 \exp\left(-\frac{n}{\tau_1}\right), n = 0,1,2,.... \tag{12}$$

where σ_0 is the value of σ at the initiation of the SOM algorithm, and τ_1 is the time constant. Correspondingly, the topological neighbourhood assumes a time-varying form of its own, as shown by equation given below:

$$h_{j,i(x)}(n) = \exp\left(-\frac{d_{j,i}^2}{2\sigma^2(n)}\right), n = 0,1,2,....... \tag{13}$$

where $\sigma(n)$ is defined by equation (12). Thus, as time n (i.e., the number of iteration) increases, the width $\sigma(n)$ decreases at an exponential rate, and the topological neighbourhood shrinks in corresponding manner. Hence, $h_{j,i(x)}$ is referred to as neighbourhood function.

3.2.3 Adaptive Process

The last process in the self organized formation of a feature map is a synaptic adaptation process. For the network to be self organized, the synaptic weight vector w_j of the neuron j in the network is required to be modified in relation to the input vector x. The Hebbian hypothesis [5] in its basic form is unsatisfactory due to the fact that change in connection occurs in one direction only, which finally drives all the synaptic weights into saturation. The Hebbian hypothesis is modified with the inclusion of forgetting term $g(y_i)w_j$, where w_j is the synaptic weight vector of neuron j and $g(y_j)$ is some positive scalar function of response y_j and

$$g(y_j) = 0 \;\; for \; y_j = 0 \tag{14}$$

The change to the weight vector of neuron j in the lattice can be expressed as:

$$\Delta w_j = \eta y_j x - g(y_j) w_j \tag{15}$$

where η is the learning rate parameter of the algorithm. The first term on the righ-hand side of equation (15) is the Hebbian term and the second term is the forgetting term. To satisfy the requirement of equation (14), linear function $g(y_j)$ has been expressed as:

$$g(y_j) = \eta y_j \tag{16}$$

equation (15) may be further simplified by setting

$$y_j = h_{j,i(x)} \tag{17}$$

using equation (16) and equation (17), equation (15) is expressed as:

$$\Delta w_j = \eta h_{j,i(x)}(x - w_j) \tag{18}$$

Finally, using discrete-time formalism and taking the synaptic weight vector $w_j(n)$ of neuron j at time n, the updated weight vector $w_j(n+1)$ at time $(n+1)$ is defined by [7]

$$\Delta w_j(n+1) = \Delta w_j(n) + \eta(n) h_{j,i(x)}(n)(x - w_j(n)) \tag{19}$$

which is applied to all the neurons in the lattice that lie inside the topological neighbourhood of winning neuron i. Equation (19) has the effect of maximizing the synaptic weight vector w_i of winning neuron i towards the input vector x. Upon repeated application of training data, the synaptic weight vectors tend to follow the distribution of input vectors due to the neighbourhood updation. The algorithm therefore leads to the topological ordering of the feature map in the input space in the sense that neurons adjacent to the lattice will tend to have similar synaptic weight vectors.

3.3 Multilayer Perceptron

Multi-layer perceptron (MLPs) with error-back propagation training have been successfully applied in a variety of pattern recognition problems [2,4,14]. They have good discrimination capability and can generate complex nonlinear decision boundaries. All the properties are very useful for speech recognition and phoneme segmentation. MLP may have any number of hidden layers, although additional hidden layers tend to make training slower, as the terrain in weight space becomes more complicated. To train the MLP, a modified version of well known Back Propagation Algorithm [5] has been used. To avoid the oscillations at the local minima a momentum constant has been introduced which provides optimization in the weight updating process. The algorithm is detailed below:

3.3.1 Initialization

The weights of each layer have been initialized to random number lies between -1 to +1.

3.3.2 Forward computation

In the forward pass the synaptic weight remain unaltered throughout the network and *functional signal* of the network is computed neuron-by-neuron basis. The induced local field $v_j^{(l)}(n)$ for neuron j in layer l which is due to the functional signal produced by neurons of layer (l-1) is given by [14]

$$v_j^{(l)}(n) = \sum_{i=0}^{m_0} w_{ji}^{(l)}(n) y_i^{l-1}(n) \tag{20}$$

where m is the total number of inputs, excluding bias applied to neuron j. The synaptic weight w_{j0}, corresponds to fixed input y_0=+1, equals the bias b_j applied to neuron j. Hence the functional signal appearing at the output neuron j of layer l is expressed as

$$y_j^{(l)} = \psi_j(v_j(n)) \tag{21}$$

If the neuron j is in the first hidden layer

$$y_j^{(0)} = x_j(n) \tag{22}$$

where $x_j(n)$ is the j^{th} element of the input vector. If on the other hand, network j is in the output layer of the network, and L the depth of the network, then

$$y_j^{(L)} = o_j(n) \tag{23}$$

where $o_j(n)$ is the j^{th} element of the output vector. The output is compared with the desired response $d_j(n)$, obtain the error signal $e_j(n)$ for the j^{th} output neuron

$$e_j(n) = d_j(n) - o_j(n) \tag{24}$$

3.3.3 Backward computation

The backward pass starts at the output layer by passing the error signal leftward through the network, layer by layer, and recursively computing the δ (i.e. the local gradient) for each neuron as follows:

$$\left.\begin{aligned} \delta_j^{(l)}(n) &= e_j^{(L)}(n)\varphi'(v_j^{(L)}(n)), \\ &\quad \text{for neuron j in output layer L} \\ &= \varphi'(v_j^{(L)}(n))\sum_k \delta_k^{(l+1)}(n) w_{kj}^{(l+1)}(n), \\ &\quad \text{for neuron j in hidden layer l} \end{aligned}\right\} \tag{25}$$

where $\varphi_j'(.)$ denotes differentiation with respect to the argument. The weight updation is taking place in accordance with the following rule:

$$\begin{aligned} w_{ji}^{(l)}(n+1) &= \\ &w_{ji}^{(l)}(n) + \alpha[w_{ji}^{(l)}(n-1)] + \eta\delta_j^{(l)}(n) y_i^{(l-1)}(n) \end{aligned} \tag{26}$$

where η is the learning rate and α is momentum constant

It has been observed that MLP based speech recognizer work better if the input and output lies between 0 – 1. Therefore, the input vector has been normalized with respect to their maximum and minimum value.

A momentum constant α has been used to avoid oscillation at the local minima. The learning rate parameter has been changed gradually with each epoch number as expressed by equation given below:

$$\eta(epochNumber) = \eta_0 \exp\left(\frac{-epochNumber}{100}\right) \tag{27}$$

where η_0 is the initial learning rate parameter.

3.4 Viterbi Algorithm

The Viterbi algorithm is a dynamic programming algorithm for finding the most likely sequence of states (normally hidden), called the Viterbi path that results in a sequence of observed events. The terms "Viterbi path" and "Viterbi algorithm" are also applied to related dynamic programming algorithms that discover the single most likely explanation for an observation.

The Viterbi algorithm was conceived by Andrew Viterbi in 1967 as an error-correction scheme for noisy digital communication links. The Viterbi algorithm operates on a state machine assumption. At any time the system being modeled is in some state. There are a finite number of states, however large. Multiple sequences of states (paths) can lead to a given state, but one is the most likely path to that state, called the "survivor path". This is a fundamental assumption of the algorithm because the algorithm will examine all possible paths leading to a state and only keep the one most likely. This way the algorithm does not have to keep track of all possible paths, only one per state.

Another key assumption is that a transition from a previous state to a new state is marked by an incremental metric, usually a number. This transition is computed from the event.

Another key assumption is that the events are cumulative over a path in some sense, usually additive. So the crux of the algorithm is to keep a number for each state. When an event occurs, the algorithm examines moving forward to a new set of states by combining the metric of a possible previous state with the incremental metric of the transition due to the event and chooses the best. The incremental metric associated with an event depends on the transition possibility from the old state to the new state. Additionally, in many cases the state transition graph is not fully connected. It must enter the stop state. After computing the combinations of incremental metric and state metric, only the best survives is kept and all other paths are discarded. There are modifications to the basic algorithm which allow for a forward search in addition to the backwards one described here.

3.4.1 Viterbi Algorithm for Isolated word Recognition

To find the best matching word by comparing input utterance with speech models in memory is very important and major task in speech recognition system. The Viterbi algorithm is an efficient technique to perform this task. Viterbi scorer computes the probability of generating the test word with each word model, and chooses one word model that gives the highest probability as the recognized word. Given an observation symbol sequence $\{O_1, O_2, O_3, \ldots, O_t\}$, the following logarithm-integer version of the original Viterbi algorithm is performed for each word model λ^v ($1 \le v \le V$, V: number of reference words) [3,6].

Initialization:

$$S_1(1) = b_j(O_1), S_1(i) = -\infty, 2 \le i \le N \tag{28}$$

Recursion:

$$\begin{aligned} S_t(j) = Max \left\{ S_{t-1}(i) + a_{ij} + b_{ij}(O_t) \right\}, \\ 1 \le i \le N,\ 2 \le t \le T,\ 1 \le j \le N \end{aligned} \tag{29}$$

Termination:

$$P_v = S_T N \tag{30}$$

where N denotes the number of states in the model, and $\{a_{ij}\}$ and $\{b_{ij}(O_t)\}$ are obtained from the state transition and the output probabilities respectively by taking logarithmic transformation followed by normalization. After computing P_v for all reference words, one word with the highest P_v is selected as the recognized word. Since multiplications in the original Viterbi scoring procedure are time consuming operations, they are converted to additions by taking logarithms of the probabilities, and underflow can be avoided efficiently.

$$a_{ij} = u.\log(\alpha_{ij}) \tag{31}$$

$$b_{ij}(O_t) = u.\log(\beta_{ij}(O_t)) \tag{32}$$

where $u.\log(x) = C.(\log_{10} x) + \Delta$ (33)

In the equation (33), the exact values of C and Δ must be determined so as to maximize the dynamic range of the transition metrics and the output metrics.

3.4.2 Viterbi Training

The MLP output gives the probability estimation for each phoneme. The Viterbi decoding algorithm will be used to find the likelihood of the best path (state sequences) for each model. Further, since the most probable transition was used at each step, backtracking can be made which gives the corresponding state sequence. Conceptually, segmentation of training data with a known model transcription is the same as recognition, except in the former case there is no alternate model sequences to consider.

Since emission probability is obtained for each frame and state category, each of these is used in a process that is often called Viterbi alignment. In this process, dynamic programming has been done, essentially using the one-pass method, in which the local distances are $-\log(y_j \mid q_\ell)$, and where there are transition costs $-\log(q_\ell \mid q_k)$ for hypothesizing transitions from states k to ℓ. Unlike the recognition scenario, the only model sequences that are considered are the ones associated with the known word sequence. All the word models together can be considered as a single model for the entire utterance in this case, and there is only one to be evaluated. Backtracking can be done since the best previous state can be preserved for each frame, and so the best state sequence can be found. Additionally, since only one model sequence need to be evaluated, often it is not necessary to use elaborate data structure for this process – the distance and backtracking information can be held in complete matrices, since the storage is not prohibitive as it would be in the recognition case.

The state sequence that is found through the backtracking procedure is considered to be an alignment of the states with the feature vectors. In the next step, transition and emission probabilities are removed assuming that the state sequence is correct.

Finally, the solution must be accessed. This can be done by looking at the changes in the global likelihood and setting some threshold on the improvement. Another approach is to test for convergence of the segmentation by counting the number of phonemes for which the state label has been changed.

4. EXPERIMENT AND PERFORMANCE EVALUATION

All experiments were conducted using a data set of 300 search queries from 15 male and 15 female speakers. The search queries are made on the basis of availability and price on 10 items of a typical supermarket. Each speaker participates in only one recording session. Each speaker uttered 10 search queries out of which 6 has been used for training the system and remaining 4 has been used for testing. The sound from the speaker has been directly digitized in WAV PCM format and sampling at 16 KHz frequency with 16 bit mono quantization. The digitized speech signal is blocked into frame of 30ms with a frame rate of 10ms. 32 Mel Frequency Cepstral Coefficients has been obtained from each frame. Their first order derivatives are also obtained. In order to reduce the computational cost, some of the less useful cepstral coefficients can be discarded. In the present study cepstral coefficient from 6-25 and its first order derivative has been considered. Thus, the feature set for each frame consists of 36 components.

For the phoneme segmentation block, a new feature set has been derived from the original MFCC based feature set. Inter-frame differences between two consecutive frames have been calculated and it is normalized between -1 to +1. We call it differential MFCC (DMFCC). Inter-frame differences obtained from 5 consecutive frames have been considered as a single input feature block for the phoneme segmenter. Each block consists of 144 DMFCC parameters. An MLP-based phoneme segmenter with 144 input nodes, 22 hidden nodes and one output node has been used for the segmentation of the phoneme. The output node will return 1 if the 3rd frame of the original feature vector is a phoneme boundary, in all other cases it will return 0. In each pass the feature vector will slide by one frame.

After identifying the phoneme boundaries, the next step is to phoneme-wise blocking the frames. The frames which belong to a particular phoneme are grouped into a single block. The feature vectors that belong to a particular phoneme are then normalized between -1 to +1. The normalized feature vectors are then clustered into five clusters using self-organized map (SOM). The output of the SOM is the cluster center of the five clusters each having 36 components. It has been observed the phoneme segmenter gives a 91.24% accuracy for 10 ms duration.

An MLP based speech recognizer with 180 input nodes, 32 output nodes and varying number of hidden nodes and layers has been constructed. Initially one hidden layer is used and the number of hidden nodes has been kept at 20 to provide 3600 connections between the nodes of input and hidden layer and 640 connections between the nodes of hidden and the output layer. The Enhance Back Propagation algorithm has been used to train the recognizer.

Gradually, the number of hidden nodes and layers has been increased. With the increasing number of hidden nodes and layers, better performance in terms of recognition accuracy has been obtained. But, with the increasing number of hidden layers and nodes, the time required for convergence is also increased. Considering all parameters, it has been noticed that MLP with 3 layers (two hidden and one output) give the optimal performance for the recognition of 32 Assamese phonemes. In the present study, a recognizer with 40 and 20 nodes in the first and second hidden layer respectively has been used. The performance of the recognizer is evaluated for each phoneme and result is summarized in the Table – (1).

Table (1): Recognition accuracy of MLP-based speech recognizer for Assamese phonemes

Category	**Sub Category**	**Recognition Accuracy (in %)**	**Average Recognition Accuracy (in %)**
Vowel	Vowel	92.21	92.21
Consonant	Nasal	84.23	85.72
	Voiced Fricative	91.69	
	Unvoiced Fricative	84.90	
	Voiced Stop	89.21	
	Unvoiced Stop	79.82	
	Roll	85.88	
	Glide	84.32	
Overall Recognition Accuracy			**88.97**

The output of the MLP-based recognizer is a sequence of phonemes. Viterbi search technique has been applied to find the keyword associated with the sequence of these phonemes. The database consists of search queries made on the basis of 10 items of a typical supermarket. For each item there are two types of search queries – availability and price. In the present study item name along with availability and price are considered as keyword. Thus, the queries consist of 12 keywords. Each search query is associated with two keywords. Viterbi alignment has been done for these 12 known keywords. To train the Viterbi search network, also known as Viterbi alignment, the keywords are manually isolated from the search query. The isolated keywords are now taken as input. The phoneme segmenter segments the phoneme and the MLP based phoneme recognizer recognizes the phonemes associated with each keyword. Viterbi alignment has been done with these known keywords. After training the network with these known keywords and phoneme sequences, the network is used for finding the keyword associated with the sequence of phoneme generated by the MLP based phoneme recognizer from the input search query. The Viterbi search tries to find the keyword that matches with the highest number of phonemes in the sequence. If an unmatched condition reached, the algorithm automatically slide one phoneme left in the sequence and restart the process.

It has been observed that the recognition accuracy of the system depends on the position of the keyword in the search query. If the keyword appears in the beginning of the search query, the recognition accuracy is higher than that if it appears in the middle. Further, it has been observed that performance of the Viterbi based keywords recognizer depends highly on the performance of the phoneme segmenter and phoneme recognizer. Therefore, in the present study, to evaluate the performance of the Viterbi based keyword recognition, only those cases have been considered where the phoneme recognizer generate the correct sequence of phonemes for a known input. The performance of the keyword recognizer is summarized in the Table – (2).

Table – (2): Recognition accuracy of the keyword recognizer

Position of the keyword in the Query Phrase	**Recognition accuracy (in %)**
Beginning	95.82
Middle	88.67
End	90.78
Average recognition accuracy	**89.76**

Further, the performance of the system is evaluated in a blind mode. It has been observed that the system can recognize the keywords associated with a spoken search query with an accuracy of 74.67%.

5. CONCLUSION

In this article, a new approach has been proposed and evaluated for the identification of search key associated with a spoken search query. The performance of the system has been evaluated on the basis of a database of a typical supermarket. The proposed system has three major components – phoneme segmenter, phoneme recognizer and keyword recognizer. It has been observed that the overall performance of the system depends on the performance of these three independent components. In the present study MLP has been used as phoneme segmenter with a feature set based on the difference of two consecutive frames. It has been observed that the phoneme segmenter give an accuracy of 91.24% for 10 ms duration. For a correctly segmented phoneme-wise feature set, the phoneme recognizer gives a recognition accuracy of 88.97%. The third of the components that is the Viterbi based keyword recognizer can recognize the keyword with an accuracy of 92.76% for correctly recognized phoneme sequence. The system has a blind mode recognition accuracy of 74.67%, which justify the fact that error occurred in each component has a cumulative effect on the overall accuracy of the system. In the present study, it has been observed that by component-wise improving the performance, especially in the first two components, i.e., phoneme segmenter and phoneme recognizer, the overall performance can be improved.

6. REFERENCES

[1] Ahad, A., Fayyaz, A. and Mehmood, T. 2002. Speech recognition using multi-layer perceptron, Proceedings of IEEE Students Conference (ISCON-02), Vol. 1, 103 – 109.

[2] Box, G.E.P, Jenkins, G.M. and Reinsel, G.C. 1994. Time Series Analysis – Forecasting and Control, 3rd Edition, Englewood Cliffs, NJ, Prentice-Hall.

[3] Buniet, L and Fohr, D. 1995. Continuous Speech Segmentation with the Gamma Memory Model, Proc. of EUROSPEECH'95, 1685-1688.

[4] Carpenter, C.A. and Grossberg, S. (Eds.). 1992. Neural Network for Vision and Image Processing, MIT Press.

[5] Gelenb, E. (Eds.). 1991. Neural Network: Advances and Applications, North-Holland, New York.

[6] Grayden, D.B. and Scordilis, M.S. 1994. Phoneme segmentation of Fluent Speech, Proc ICASSP, 73-76.

[7] Kohonen, T. 1995. Self-Organized Maps, Springer-Verlag.

[8] Schwatz, R. and Makhoul, J. 1975. Where the Phoneme Are: Dealing with Ambiguity in Accoustic-Phonetic Recognition, IEEE Trans. ASSP, Vol. 23, 50-53.

[9] Sodani, M., Nitsuwat, S. and Haruechaiyasak, C. 2010. Thai Word Recognition Using Hybrid MLP-HMM, International Journal of Computer Science and Network Security, VOL.10 No.3, 103-110.

[10] Suh, Y. and Lee, Y. 1996. Phoneme Segmentation of Continuous Speech using Multilayer Perceptron, ICSLP 96, 1297-1300.

[11] Talukdar, P.H., Bhattacharjee, U., Goswami, C. and Barman, J. 2005. A Robust Recogniser for Assamese and Bodo Vowels using Artificial Neural Network, Proc. Int. Sym. Frontiers of Research on Speech and Music-2005, 148-152.

[12] Ting, H.N., Jasmy, Y., Sheikh Hussain, S.S. and Cheah, E.L. 2001. Malay syllable recognition based on multilayer perceptron and dynamic time warping, Proceedings of the Sixth International Symposium on Signal Processing and its Applications, vol. 2, 743 – 744.

[13] Weinsterin, C.J., McCandless, S.S., Mondehin, L.F. and Zue V.W. 1975. A System for Acoustic Phonetic Analysis of Continuous Speech, IEEE Trans. ASSP, Vol. 23, 54-67.

[14] Zeidenberg, M. 1990. Neural Network Models in Artificial Intelligence, E.Horwood, London.

[15] Zue V.W. 1985. The Use of Speech Knowledge in Automatic Speech Recognition, Proceedings of the IEEE, Vol. 73, 1602-1615.