JJEE Volume 9 | Number 4 | December 2023 | Pages 509-536

Jordan Journal of Electrical Engineering

ISSN (print): 2409-9600, ISSN (online): 2409-9619

Homepage: jjee.ttu.edu.jo

A Hidden Markov Model-Based Speech Recognition System Using Baum-Welch, Forward-Backward and Viterbi Algorithms

Benjamin Kommey[1*], Ernest O. Addo[2], Elvis Tamakloe[3]

[1, 2, 3] Computer Engineering Department, College of Engineering, Kwame Nkrumah University of Science and Technology, Kumasi, Ghana

E-mail: bkommey.coe@knust.edu.gh

Received: February 09, 2023 *Revised: April 27, 2023* *Accepted: May 08, 2023*

Abstract— Speech is the most complex part or component of human intelligence and for that matter speech signal processing is very important. The variability of speech is very high, and this makes speech recognition difficult. Other factors like dialects, speech duration, context dependency, different speech speed, speaker differentiation, environment and locality all add to the difficulty in speech processing. The absence of distinct boundaries between tones or words causes additional problems. Speech has speaker dependent characteristics, so that no one can reproduce or repeat phrases in the same way as another. Nevertheless, a speech recognition system should be able to model and recognize the same words and phrases absolutely. Digital signal processors (DSP) are often used in speech signal processing systems to control these complexities. This paper presents a Hidden Markov Model (HMM) based speech signal modulation through the application of the Baum-Welch, Forward-Backward and Viterbi algorithms. The system was implemented using a 16-bit floating point DSP (TMS320C6701) from Texas instruments and the vocabulary was trained using the Microsoft Hidden Markov Model Toolkit (HTK). The proposed system achieved about 79% correct word recognition which represents approximately 11,804 correct words recognized out of a total of 14960 words provided. This result indicates that the proposed model accuracy and speaker independent system has a very good evaluation score, and thus can be used to aid dictation for speech impaired persons and applications in real time with a 10 ms data exchange rate.

Keywords— Speech recognition; Digital signal processing; Hidden Markov model; Baum-Welch algorithm; Forward-backward algorithm; Viterbi algorithm.

1. INTRODUCTION

The term speech recognition is used when the acoustic signals received by one or more microphones or telephones are converted into a single word or sentence. Thus, speech recognition can also be described as an automated process, with a machine converting an acoustic signal into equivalent text form. Natural language has some characteristics or features like speech which is serial, meaning that the speaker can only speak one word at a time, but not several words at the same time. A language is relatively slow, and humans can read about 350-500 words faster than listening (between 175-225 words), according to a study by [1]. This means that listening is slower than reading. Language needs articulation because it must be spoken and is therefore public and speech is an acoustic phenomenon of air pressure that changes over time. A distinction is made between speech recognition and speaker recognition. Speech recognition means the ability to recognize the word that is being said. If the question "who said that?" can be answered from the word, then one speaks of speaker detection or recognition.

* Corresponding author

This work deals with speech modelling for recognition, with explanations of speaker recognition being given at various points. The recognition of spoken language is undoubtedly one of the most difficult and at the same time most exciting tasks in speech signal processing. A speech modeler or recognition system consists mainly of a signal representation where the digitization takes place, an acoustic model to extract the speech properties and finally a word or language model (See Fig. 1).

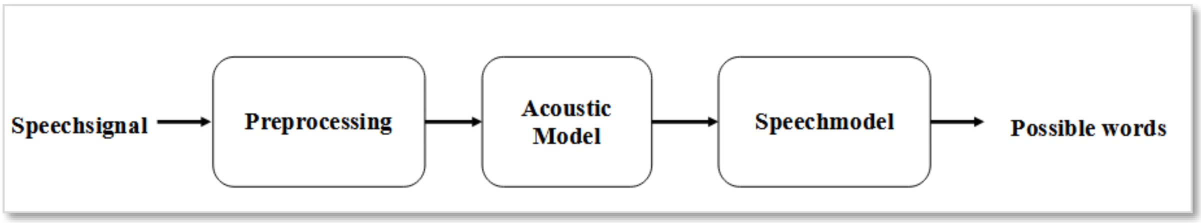


Fig.1. Speech recognition process.

As shown in Fig. 1, the recognition process takes place in three steps. In the first step, the analog voice signal is digitized, i.e., converted into a sequence of binary numbers. The computer can only process the speech signal in this form (digital). In the second step, the information required for a clear description of the speech (time, frequency, and speech intensity) is extracted from the given speech signal. This helps to reduce the amount of data. To do this, the speech signal is divided into short time segments (frames), with the speech intensity being determined for around 20 ms in each time segment. The determined values are seen as language features and together form a feature vector. In the final step of the recognition process, the phonemes or words corresponding to the speech signal are determined. There are various methods for this phoneme or word determination, which are briefly described in the following sections. There are various program packages for setting up a speech recognition system such as the CSLU Speech Toolkit and Hidden Markov Model Toolkit (HTK). The HTK toolkit was used in this work and is described in the following subsection [2, 3].

Speech recognition systems can be divided into three points; speaker dependency, type of utterance (single word or continuous) and range of vocabulary. Depending on the speaker dependency, a distinction is made between a speaker-dependent system and a speaker-independent system. Speaker-dependent systems, whereby the system can only recognize the words of a speaker. Only speech signals from this speaker are stored in the speech database. When recognizing, the spoken word is compared with the stored word and recognized. The process of language training is later explained. The trained person’s spoken words can be recognized up to 98% with such systems, but the downside is that only the person who trained the system can use it. The system is used in dictation machines and voice-controlled telephone dialing systems. On the other hand, in a speaker-independent system, the system can recognize words from any speaker. The recognition process is the same, but here speech signals from different speakers are used for training. Speaker-independent systems are more flexible than speaker-dependent systems because the system responds to a specific word rather than a specific person’s voice. On the other hand, a speaker-independent system is more error-prone. The complexity of both systems increases with the number of words that the system is supposed to recognize. The recognition rate is strongly influenced by background noise, the speaker's voice (clear, fast, or slow), and the difficulty of distinguishing acoustically similar words, such as the English words "no" and "go". The speaker-adaptive system is like the speaker-dependent system with the difference that the system can adapt to new speakers during recognition and that it is initially trained speaker-independently. According to the type of statement, the following is to be broken down; word-continuous system: in which the user speaks normally with no pause between words. Compound word system: a single word recognition system without a clear pause (pause < 250ms). However, the word must be spoken clearly and distinctly to be recognized. Speech recognition systems already exist to recognize compound numbers, but none of them can recognize compound words. Single word system: this system requires a short pause between words (pause > 250ms), such as entering the phone number 23567 (two-pause-three-pause-call-pause-six-pause-seven) [1].

The vocabulary range is classified as follows; Small Vocabulary: System of a hundred or fewer words. Medium-sized vocabulary: system with more than a hundred words but less than a thousand words. Large vocabulary system: One thousand (1000) word system to an unlimited number of words. Speaker-independent systems support a vocabulary range of up to one hundred words, most mainly recognized numbers and some commands like "yes, no, stop". However, there are also speaker-independent systems that support larger vocabulary ranges, e.g., Nortel's Flexible Vocabulary Recognition (FVR) technology system. The recognition rate of speech recognizer is influenced by several factors. Some of these factors are: Physical characteristics of the speaker, location and environment of the speaker (i.e., regional dialects, pronunciation), age of the speaker (a child, teenager or an elderly person), ethnicity (e.g., English or American) and gender-specific speaker distinction (male and female), speech rate (slow, fast, or too fast), word stress (strong or light), background noise, delays, context of the phonemes (e.g., pay and pain), poor articulation (words slurred), current state of the speaker (stress, anxiety, etc.), reception or recording quality, phoneme confusion (e.g., go and no, two days and today, cents and sense). Thus, this project aimed to improve the word recognition rate regardless of the afore mentioned constraints.

2. RELATED WORKS

In [4], a real-time control of a mobile robot using HMM-based speech recognition system was presented. This proposed system developed an HMM with separate word recognition system for a mobile robot car and has real-time control. The control design of the mobile robot employed Mel-Frequency Cepstral Coefficients (MFCC) as its features. About 270 speech commands were sampled from 54 different people and applied directly to a series of mathematical operations and a total of 12 cepstral coefficients were derived. A database was then generated by these 12 cepstral coefficients and the HMM model was trained and tested accordingly. Although this system provides about 94% recognition success rate of test commands, its performance depreciates appreciably in noisy environment. Thus, the recognition success rate is highly dependent on the environment of operation.

The problem of assigning probability values for HMM with some limited inputs by experts has resulted in some inaccuracies in other cases related to the same domain. In [5], a proposed optimization of Hidden Markov Model with Gaussian mixture densities for Arabic speech recognition was presented. This employed a general approach based on Genetic

Algorithms (GAs) to find the most suitable probability values for the HMM with more accuracies in variety of cases than its initially assigned probability. Even though this GA/HMM model provided an increase in recognition rate from 3.2% to 16.3%, it nonetheless requires higher performance improvements, a well-structured dataset, and an enhanced database size. The response time was also not provided to ascertain its real time applications. In a segment-based speech recognizer, the use of segmentation networks and segment-based features complicates probabilistic modelling due to alteration of the sample space of all possible segmentation paths and feature observation space. Regarding this challenge, a Baum-Welch training algorithm for segment-based speech recognition was developed in [6]. This algorithm achieved a relative error reduction of 37% and 5% on the training set and test set respectively compared to Viterbi training models when tested. An overall word error rate (WER) of 7.6% was also obtained with this model when combined with a duration model. Although a slight improvement in the word error rate was achieved, more training is required since WER improves significantly with training. Additionally, training with this model is slow due to its several iterations through the training data.

Several contributions have been made to improve the use of HMM as a probability model in statistical learning and series data processing [7-9]. In relation to this, speech processing as applied to Internet of Vehicles (IoV) based on Hidden Markov Model and Vector Quantization (VQ) techniques was proposed in [10]. This primarily comprises of voice signal pre-processing, speech recognition and speaker recognition. The speech model was developed using MFCC method for extraction of voice features, HMM for long term statistics and a Viterbi algorithm to search for the best sequence in the model for speech recognition. A Linde-Buzo-Gray (LBG) algorithm in VQ was used to train the speaker model and a cosine similarity employed to achieve the function of speaker recognition. The voice processing results are transmitted to the IoV system based on the recognition results; the system determines whether to execute the voice command. This proposed system yielded results with only 50 test samples. However, this number of test samples is extremely small to fully evaluate the system. The effects on results due to environmental constraints were not discussed. The proposed method expressed by [11] employed a modified Forward-Backward Re-estimation algorithm to recognize speech patterns. A TI46 database with a total database size of 4160 utterances was used for the experiment. Among these, 1600 samples were trained, and the remaining was used for testing of ten English words. Additionally, a vector of 12 Linear Predicting Coding Cepstrum coefficients was derived and given as an input to vector quantization (VQ) to find for each class code-word. Each continuous observation vector is mapped into a discrete code-book index with a VQ codebook, and the vector quantized values provided to a Left-Right model for modeling. Based on this, likelihoods are calculated for matching and recognizing the English words. An increase in recognition accuracy of 4% in the 3-state model was obtained with the proposed modified Forward-Backward Re-estimation algorithm when compared with the conventional Forward-Backward Re-estimation algorithm using Left-Right model. However, in terms of 5-state models, the conventional algorithm has higher recognition accuracy than the modified algorithm.

Regarding Hidden Markov Models (HMMs), many techniques and algorithms have been developed to help maximize its use in achieving very efficient computation in variety of fields including speech recognition [12-17]. Based on this premise, an isolated spoken word recognition system using HMM was designed and implemented in [18] on a 3R robotic arm. A variety of command words were recorded by human voice and the results compared with a Mel Frequency Cepstral Coefficient (MFCC) and fundamental frequency as feature extraction methods. The HMM was trained by utterance of the command to constitute supervised and unsupervised techniques. The training of the data with diagonal covariance matrix was utilized for all states and for each word collected, a Baum-Welch algorithm runs 15 iterations for each command. Although this system is very practical to implement, training with this model is slow due to its several iterations through the training data.

3. MATERIALS AND METHODS

3.1. System Theories

Speech recognition starts with the preparation of the speech signal, which includes 3 steps: digitalization, pre-emphasis, windowing and characteristics or feature extraction. Digitalization by way analogue to digital conversion where the speech recorded with the microphone is analogue and must be converted into a digital signal for further processing with the computer. The pre-emphasis is the first of the feature extraction. The speech signal is amplified as a function of frequency, and areas with higher frequency components are boosted more than areas with smaller frequency components. At this point, there is now a discrete signal that allows the use of a discrete filter. In speech signal processing, a first order FIR filter is used for this task with the function in Eq. (1).

$$H(z) = 1 - az^{-1} \tag{1}$$

where $H(z)$ and az^{-1} represents the transfer function and denote the filter coefficient in reference to the z plane.

The speech signal is broken down into temporally short sections (frames). It is assumed here that the speech signal is stationary (unchanged) in short periods of time. In general, a duration of approximately 25 ms is chosen. The windowing creates a series of individual observations (frames) $s_t(n)$, t = 1, 2, … T, each representing a short interval of the original signal. A discrete Fourier transformation (DFT) is then carried out for these short-term intervals and later transformed into the frequency domain. Calculations are performed by increasing the window by a certain number of individual observations shifted in time steps and then multiplied by the window function.

3.1.1. Fourier Analysis

The Fourier transform (FFT) is a mathematical method used to calculate frequency representation from a given time representation of a signal. The Fourier analysis is based on the idea that the signal corresponds to a complex oscillation that is made up of individual sine and cosine components. If the speech signal to be analyzed contains the properties of both an excitation function and the vocal tract filter, the output signal obtained with the Fourier analysis also shows the spectral properties of the excitation function and the vocal tract filter. In this analysis, a time-continuous and periodic signal is transformed into a discrete spectrum in the frequency domain. For a signal g(t) the following applies (Eq. (2)):

$$G(\omega) = \int_{-\infty}^{\infty} g(t)e^{-j\omega t}dt \tag{2}$$

If the signal to be transformed is a time-discrete sequence, one speaks of a discrete Fourier transformation. To be able to calculate the spectral components, all individual observations are subjected to a Fourier transformation. For a time-limited signal $s_t(n)$ of length N, the following applies regarding the FFT:

$$F\{s_t(n)\} = S_t e^{(j\omega)} = \sum_{n=0}^{N-1} s_t(n) e^{(-j\Omega n)} \quad (3)$$

where ω represents the continuous frequency axis. Ω is the normalized angular frequency in Eq. (3).

3.1.2. Linear Prediction Coefficient Analysis

Linear prediction is used in information processing to reduce data for the transmission, in which only a resulting prediction error is transmitted, from which the output signal can be developed again with a corresponding receiver. In speech recognition systems, this technique is also used in the modeling of a simple system for speech recognition or speech production. As the name suggests, this is about predicting the output signal value (s(n)) from the linear combination of the input signal values (s(1) +s(2)+…+s(n)) as shown in Fig. 2.

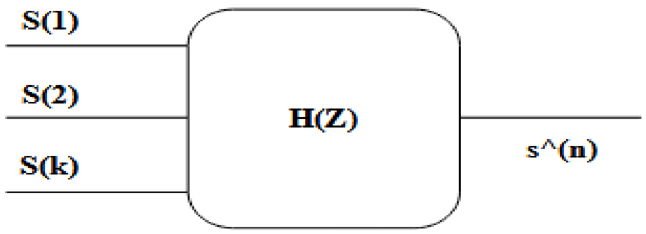


Fig. 2. Linear prediction coefficients analysis.

It is mathematically formulated as:

$$s(n) = \sum_{k=1}^{K} a_k s(n-k) \quad (4)$$

The transfer function results in the z-plane is:

$$H(z) = \frac{1}{\sum_{i=0}^{p} a_i z^{-1}} \quad (5)$$

This is an all-pass filter with the filter coefficients a_i, (a_0 = l) and p as the number of poles. The filter coefficients are selected in such a way that the prediction errors, summed over a window (Hamming window, in Eq. (6)) becomes minimal.

$$\omega(n) = \begin{cases} 0.54 - 0.46 cos\left(\frac{2\pi n}{M}\right) 0 \leq n \leq M \\ 0 otherwise \end{cases} \quad (6)$$

This is a convolution or autocorrelation method. Further mathematical details are omitted, and reference is made to [19] and [21]. The following example shows how this autocorrelation method is used in the HTK [3]. Given a window signal $s_n(n)$ with $\{s_n(n)$, $n=1…N\}$ in Eq. (4), the p+1 term can be calculated using the autocorrelation function in Eq. (7).

$$AKF(i) = r_i = \sum_{j=1}^{N-i} S_i S_{j+i} \quad (7)$$

Where (i = (0, p)). The filter coefficients are calculated recursively with several so-called reflection coefficients $\{k_i\}$ of a similar acoustic signal and the prediction factor E (E= r_0, at the beginning) being calculated. If a filter of order i-l and the rejection coefficients $\{k_j^{i-1}\}$ and filter coefficients $\{a_j^{i-1}\}$ are given, a filter of order i can be calculated in three steps using the Eqs. (8), (9), (10) and (11).

- Calculation of the new reflection coefficients

$$k_j^{(i)} = k_j^{(i-1)} \tag{8}$$

and for j= 1, …. i-1

$$k_i^{(i)} = \frac{\left\{r_i + \sum_{j=1}^{i-l} a_j^{(a-1)} r_{i-j}\right\}}{E^{(i-1)}} \tag{9}$$

- Prediction energy update

$$E^{(i)} = \left(1 - k_i^{(i)} k_i^{(i)}\right) E^{(i-1)} \tag{10}$$

- Calculation of the new filter coefficients

$$a_j^{(i)} = a_j^{(i-1)} - k_i^{(i)} a_{i-j}^{(i-1)} \tag{11}$$

$forj = 1, \ldots, i-1 \wedge a_i^{(i)} = -k_i^{(i)}$

The process is repeated from i = 1 to the desired filter order i = p.

3.1.3. Filter Bank Analysis

Filter bank Analysis is an alternative to Linear Prediction Coefficient Analysis (LPC). It is used to determine the spectral properties of a signal. With filter banks, it is possible to split an input signal into several sub-band signals and process the individual bands separately. The signals are broken down into narrow-band frequency bands, after which it can be under-sampled and over-sampled again after signal processing. One advantage of sub-band decomposition is the possibility of performing noise reduction tailored to each individual frequency band. The filter bank can be divided into an analysis and synthesis section, with the analysis section consisting of bank limit filters and the reduced sampling rate. The individual partial band signals are reconstructed from the synthesis part. For a filter bank with k channels of the same bandwidth, it follows that the sampled values of all output signals can be taken from all r clocks. The highest under-sampled factor could be r = k if ideal band pass were present. But since real filters have finite edge steepness and the pass bands of the individual filters were too wide, there could be considerable spectral overlaps (aliasing), which would fake the calculations. The choice of r < k ensures that an uncritical range remains between the periodically interfering, under sampled spectra, in which the spectral components that come about because of non-ideal filtering have no disturbing influence. The band-pass filters are therefore referred to as anti-aliasing filters. The low-pass filter of the first channel is used to implement the band-pass filters of the remaining k-l filter bank channels. For this purpose, the impulse response of the low-pass filter is multiplied by complex factors. In the frequency range, this corresponds to the necessary frequency shift to create a band-pass filter. The complex factors can be combined and multiplied further to be represented as a Discrete Fourier Transform (DFT). If the number of filter bank channels is an integer power of 2, then the DFT can be implemented as a Fast Fourier Transform (FFT). Eq. (12) contains the Discrete Fourier Transform (DFT) analysis.

$$S(k) = \sum_{n=0}^{N-1} s(n)\, e^{(-jkn\Delta\Omega)} \tag{12}$$

where $\Delta\Omega$ is base frequency with $\Delta\Omega = \frac{2\pi}{N}$ [20].

The signal spectrum is then broken down in further steps and rescaled using the Mel-frequencies in Eq. (13):

$$Mel(f) = blog_{10}\left(1 + \frac{f}{c}\right) \tag{13}$$

Typical values according to [22] for b and c are 2600 and 700. The HTK program package for this work uses a filter bank based on a simple Fourier transform (FFT) with approximately the same resolution on the so-called Mel-scale. To reduce the computational complexity, Eq. (14) was used.

$$Mel(f) = 2595 log_{10}\left(1 + \frac{f}{700}\right) \quad (14)$$

3.1.4. Mel-Frequency Cepstral Coefficients Analysis

Mel-frequency cepstral coefficient (MFCC) analysis is the most used feature extraction in speech recognition systems. Fig. 3 depicts the MFCC processing stages.

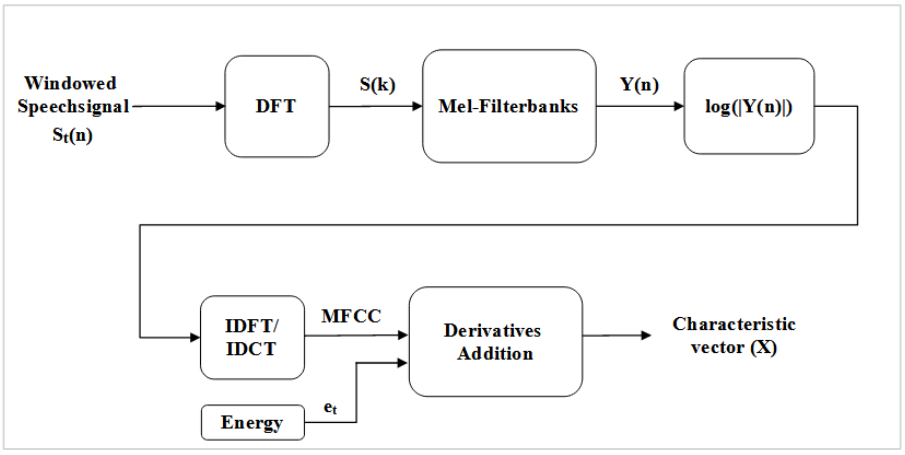


Fig. 3. MFCC processing steps.

In the expression "cepstrum," a "spectrum" is reversed, so to speak. i.e., the first syllabi of "spectrum" is read backwards to get to the "cepstrum." For example, the basic unit in spectrum is "frequency' but "quefrency" in cepstrum. As shown in Fig. 3, the cepstrum or cepstral coefficients are won from the IDFT of a logarithmic spectrum. Eq. (15) gives the mathematical expression:

$$C_n(n) = \frac{1}{2\pi}\int_{\omega=-\pi}^{\pi} log\left(Ve^{(j\omega)}\right) e^{(j\omega n)} d\omega \quad (15)$$

The calculation of the cepstra (MFCC) starts with the weighting of individual frequency ranges. Based on human speech perception, such processing is to be regarded as perceptual weighting, in which the characteristic range is amplified and irrelevant ranges from the frequency spectrum are hidden. 24 band-pass filters are generally used. The mid-band frequencies (b_m) of the Mel-filter banks are not linearly distributed over the frequency spectrum. The filter width (Δ_m) is also frequency dependent. The frequency range below 1 kHz is covered by a larger part of the filters, since that is where more prominent features of the vocal tract deformation are contained. 10 bandpass filters are used for the lower range. The distribution of Band center frequencies (b_m) is constant over the frequency range from 0 kHz to 1 kHz. Above 1 kHz, a logarithmic division is used [19].

The following are valid for the bandwidths:

$\Delta_m = 1.2 \times \Delta_{m-1}$ and $b_m = b_{m-1} + \Delta_m$

where $2\Delta_m$ is the perimeter of the mth triangular window. The triangular window shape of the bandpass is described by Eq. (14). The perceptually weighted values at the outputs of these mel-scale filters can be specified using Eq. (16). The mth filter bank output is given by:

$Y_t(m) = \sum_{k=b_m-\Delta_m}^{b_m+\Delta_m} X_t(k)U_{\Delta m}(k+b_m), 1 \leq..M$ (16)

where $U_{\Delta m}$ is the impulse response of the bandpass triangle window and

$X_t(k) = X_t\left(e^{\left(j\frac{2\pi k}{N}\right)}\right)$ (17)

with k = {0, ... N-1} as the DFT of all frames indicated in Eq. (17). From the weighted spectrum obtained through the previous processing measures, conclusions can be drawn about the influence of the vocal tract when generating the observation sequence S(k). The speech creation can be represented as a convolution of the voice-band h(k) and vocal-tract v(k) in the form (Eq. (18)):

$x(k) = v(k)*h(k)$ (18)

since the articulation of phonemes is mainly determined by the vocal tract transformation. This is of particular interest. To get the formation of the vocal tract h(k), a structure-preserving analysis must be carried out. Using this analysis, the convolution (Eq. (18)) is broken down into the components of excitation and vocal tract form using the superposition principle of the FFT and using the logarithmic laws as:

$x(k) = F^{-1}\{log(F\{v(k) * h(k)\})\} = F^{-1}\{log$ (19)

where ^v(k) and ^h(k) are the complex cepstra to the spectra. Eq. (19) can be separated by bandpass filters provided the frequency ranges do not overlap. This would be possible here because the change in the shape of the vocal tract is much slower than the harmonic excitation frequency of the vocal cords. The real cepstrum can be determined by omitting the phase information that is not of interest [19].

Taking the logarithm of Eq. (19) results in dynamic compression, which makes feature extraction more independent of dynamic fluctuations. Only real-valued coefficients result from the formation of absolute values, which means that the IDFT can be converted into the faster IDCT. The final step in determining the cepstral coefficients (MFCC), is to calculate the IDCT of the logarithm of $|Y_t(m)|$ as expressed in Eq. (20).

$Y_t^{(m)}(k) = \sum_{m=1}^{M} log\{|Y_t(m)|\}cos\left(k\left(m-\frac{1}{2}\right)\frac{\pi}{M}\right)$ (20)

In most speech recognition systems using MFCC features, coefficients $y_t^{(m)}$ for 1<k<15 are used. At k = 0, the coefficient corresponds to the energy E_0 within a single observation as in Eq. (21).

$E_0 = \sum_{n=0}^{N-1} S_t^2\,(n)$ (21)

Using the energy next to the MFCC leads to the improvement of the detection performance. Above all, the energy feature can be used to easily separate silent recording areas from those that contain spoken utterances. In speech signal processing, the rate of change of the spectrum can provide further information about the phoneme. Eq. (22) shows the cepstra as a function with respect to the signal time. The time derivative of the logarithmic spectrum often referred to as acceleration, can be obtained by dividing the cepstrum in time.

$\frac{\partial c_t(n)}{\partial t} = \frac{1}{2\pi}\int_{-\pi}^{\pi} \frac{\partial log\left(Ve^{(j\omega)}\right)}{\partial t} e^{(j\omega)} d\omega$ (22)

In the HMMs, the observations are random variables given by the Gaussian equation (Eq. (23)).

$b_j(O) = P(O|q_t = j) = \aleph\left(O, \mu_j, U_j\right), 1 \leq j \leq N$ (23)

The cepstral derivative of the weighted random variable is provided in Eq. (24):

$\frac{\partial c_t(n)}{\partial t} \aleph(O, \mu, U)$ (24)

The acceleration is given by the Gauss distribution in Eq. (25).

$\frac{\partial log(S_t(\omega))}{\partial t} \aleph(S, \mu_s, U_s), S_t(\omega) = \left|Ve^{(j\omega)}\right|^2$ (25)

To model the rate of change of the spectrum in the HMMs, the cepstral differences, called deltas, are often added to the observation vectors as seen in Eq. (26). In some systems, the short and long terms (delta δ and delta-delta Δ) of the cepstral differences are used.

$O_t = \left[\ldots, c_t(n), \ldots, \frac{c_{t+\delta}(n)-c_{t-\delta}(n)}{2}, \ldots, \frac{c_{t+\Delta}(n)-c_{t-\Delta}(n)}{2}, \ldots\right]$ (26)

where the offsets or cepstral difference windows δ and Δ are varied to model the short and long terms. According to [21], parameter estimates of the first and second cepstral derivatives can be used instead of the short and long terms of the cepstral differences. To be able to calculate the first and second cepstral derivatives, the error E is minimized and given by Eq. (27):

$E = \sum_{t=-M}^{M}[C_t(n) - (h_1(n) + h_2(n)t - h_3(n)t^2)]^2$ (27)

The window size M is comparable to the long term of the cepstral difference window Δ. In [21], Rabiner gives the optimized values of h_l, h_2, h_3 as cepstral coefficients, from which the estimates of the first and second cepstral derivatives can be calculated using Eqs. (28) and (29), respectively.

$\frac{\partial C_t(n)}{\partial t} \approx h_2(n)$ (28)

$\frac{\partial^2 C_t(n)}{\partial t^2} \approx 2h_3(n)$ (29)

3.1.5. Hidden Markov Model

An important mathematical method for describing the dynamics of stochastic processes is Markov's theory [23]. A Markov chain is a class of stochastic processes with special properties. The stochastic process is a temporal group of discrete states s_i, from the state set $\{s_1, \ldots, s_n\}$ given a random variable S. In other words, the Markov chain is a state model (Markov model) of a stochastic process in which observations of the transition set only depend on the current state. This means that the state at time t only depends on the state at time t-1 (memory-less property). Mathematically formulated, a 1-order Markov chain with N states, as shown in Fig. 4, for N = 4 is a 3-tuple (triple) given by (S, A, π).

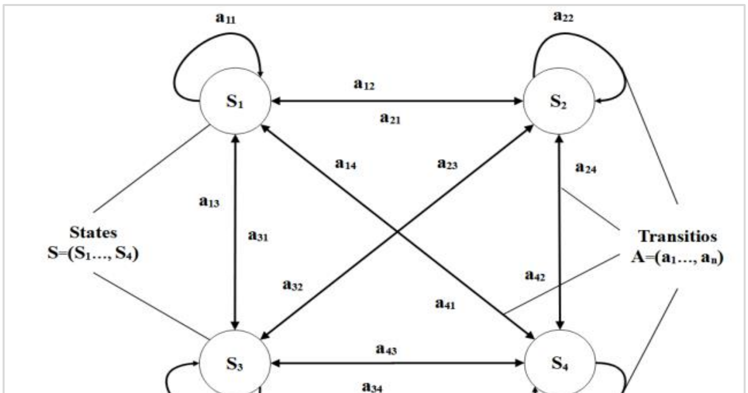


Fig. 4. The Markov chain.

S = (S_1, S_2, …, S_N) is the set of states where the system moves between well-defined states s_i. The states can be observed directly and merge directly into one another. The state transitions are independent of each other. A is the transition set, an NxN matrix of state transition probabilities, and π is an N-dimensional probability vector of a state at time t. The Markov chain represents a statistical model of a signal sequence. The observed signal sequence is based on stochastic processes. For a discrete-time Markov chain of the first order, it is represented by Eq. (30):

$$P[S_i = j|S_{t-1} = i, S_{i-2} = k, \dots] \quad (30)$$

with the transition probability in Eq. (31).

$$a_{ij} = P[S_i = j|S_{i-1} = i]1 \leq i, \wedge j \leq N a_{ij} \geq 0 \forall i \wedge j \wedge \sum_{j=1}^{N} a_{ij} = 1 \forall i \quad (31)$$

In speech signal processing, the Hidden Markov Models (HMM) are used to model the course of speech over time. Two models are distinguished, the discrete HMM and the continuous HMM. Discrete HMM requires less training (compared to the continuous HMM) and is often used for modeling in single-word recognition. Hidden Markov models are Markov models in which the exact sequence of states with which a symbol sequence was generated or accepted is not visible. What matters is the likelihood of the outcome. The HMMs can be viewed as stochastic and finite automata. Here, however, in contrast to the finite automata, the probabilities of the transitions and symbol emissions are explicitly modeled. Because the parameters are invariant over time, a Markov process of the 1st order is described by this HMM. An HMM can be described completely by a 5-tuple λ = (Q, Π, F, A, B) and consists of the following sets [23].

- State set Q = {q_1, q_2, …, q_z}
- Initial probability sets Π = {$π_1$, $π_2$, …, $π_z$} with $π_i$=P (q_i by t = 0), $\sum_{i=1}^{Z} \pi_i = 1.0$
- Set of End states F⊆ Q
- Transition set A = {a_{ij}}, a_{ij} = P (q_j by t+1 | q_i by t) and $\forall i \sum_{j=1}^{Z} a_{ij} = 1.0$
- The emission probabilities given by B = {b_j(.)}, b_j(O) = P (o by q_j | q_j by t)

When designing HMMs, the following four basic problems arise: selection of a suitable HMM basic model, probability of an observed sequence given the model, optimal state for the given observation sequence and model and optimizing the parameters of an HMM (training). General assumption of the HMM are:

- The next state depends only on the current state (memory-less property).
- The transition probabilities are time-invariant.
- The current outcome or observation is independent of previous observations.

3.1.6. Forward- Reverse-Algorithm

The goal of the forward and backward algorithm is to calculate the probability P(O | W) from the Markov chain of a word sequence W and observations O. The so-called forward variable or forward probability is:

$$\alpha_t(i) = P(o_1, o_2, \dots, o_i, q_i = i \vee \lambda),$$

where the probability of the observations set being o_1, o_2, … o_t at time t in the state I if the HMM $\lambda = (S, A, \pi)$ is given as shown in Fig. 5.

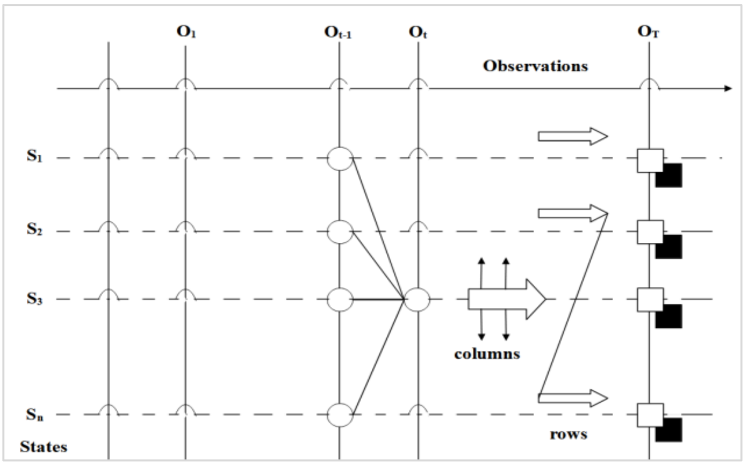


Fig. 5. Forward-backward algorithm.

The forward probabilities can be calculated recursively as follows (forward algorithm):

1. **Initialization:**

First, the forward probability is initialized as the joint probability of state i and the initial observation o_1. Eq. (32) mathematically presents the initialization as:

$\alpha_t(i) = \pi_i b_i(O_1) 1 \leq i \leq N$ (32)

2. **Recursion:**

In Eq. (33), the state j at time t can be reached from N possible and different states i at time t-1 where $1 \leq i \leq N \wedge 1 \leq t \leq T-1$.

$\alpha_t(j) = \sum_{t=1}^{N} \alpha_{t-1}(i) a_{ij} \vee b_j(O_t)$ (33)

The product is the joint probability of the joint events with the observations O = ($o_1 o_2$ … .o_t) and the achievement of state j at time t from state i at time t_l. Since the summation of the N possible states i at time t-l results in the probability of state j at time t with the accompanying observations, α_{t+1}(j) can be calculated by multiplying this variable by the observation probability $b_J(o_{t+1})$.

3. **Termination:**

$P(O|\lambda) = \sum_{i=1}^{N} \alpha_T\,(i)$ (34)

Here in Eq. (34), the probability P(O|λ) is calculated as the sum of the terminated forward probability α_T(i), where:

$\alpha_T(i) = P(o_1, o_2, \dots, o_T, q_T = i \vee \lambda)$

Analogous to the forward variable, the backward variable $ß_t(i) = P(o_{t+1}, o_{t+2}, \dots, o_T | q_t = i, \lambda)$ is defined as the probability of the observations at t+l to the end, given both the state i at time t and the HMM. In the backwards algorithm, the recursive calculation of the backward probabilities $ß_t$(i) is done backwards in time by employing Eqs. (35), (36) and (37).

- **Initialization**

$\beta_t(i) = 11 \leq i \leq N$ (35)

- **Induction**

$\beta_t(i) = \sum_{j=1}^{N} a_{ij} b_j(O_{i+1}) \beta_{t+1}(j)\, fort = T-1, T-2, \dots, 11 \leq i \leq N$ (36)

- **Termination**

$P(O|\lambda) = \sum_{i=1}^{N} \pi_i b_i(O_1) \beta_1(i)$ (37)

The backward probability follows from state i, while the forward probability is simultaneously associated with state i. It should be noted that the calculation of P(O|λ)

using forward, or backward algorithms only requires about 2NT multiplications. At any point in time t, the following relations in Eqs. (38) and (39) apply.

$$\alpha_t(j).\beta_t(j) = P(O, q_i = j \vee \lambda) \tag{38}$$

and

$$P(O|\lambda) = \sum_{i=1}^{N} \alpha_t(j)\beta_t(j) \tag{39}$$

3.1.7. Baum-Welch-Algorithm

The Baum-Welch algorithm is a variant of the Expectation Maximization (EM) algorithm for estimating HMM parameters. Unlike the forward or backward algorithm, this is an iterative method that, starting from a parameter estimate λ, calculates a new parameter estimate in each iteration step, so that Eq. (40) is achieved:

$$P_{HMM}(\lambda) \geq P_{HMM}(\lambda) \tag{40}$$

The a-posteriori transition probability $\xi_t(i,j)$ is the joint probability that the system is in state i at time t and in state j at time t+l, given the HMM and the observations. This means:

$$\xi_t(i,j) = P(q_t = i, q_{t+1} = j \vee O, \lambda) \tag{41}$$

Substituting the forward and backward variables in Eq. (41) gives rise to Eq. (42):

$$\xi_t(i,j) = \frac{\alpha_t(i)a_{ij}b_j(O_{t+1})\beta_{t+1}(j)}{\sum_{i=1}^{N} \alpha_t(i)\beta_t(i)} \tag{42}$$

For the overall observations of the system, the a-posteriori state probability $\gamma_t(i)$ for state i at time t, is given by Eq. (43) which consist of the summation of all transition probabilities.

$$\gamma_t(i) = P(q_t = i|O, \lambda) = \frac{\alpha_t(i)\beta_t(i)}{\sum_{i=1}^{N} \alpha_t(i)\beta_t(i)} = \sum_{j=1}^{N} \xi_t(i,j) \tag{43}$$

The Baum-Welch method for optimal parameter estimation for discrete-valued HMMs, is realized as follows [24], if the discrete HMM (N states, k symbols) and the training sequences are given as in Eqs. (44), (45) and (46):

$$\pi_i = \gamma_i(i) = \frac{\alpha_1(i)\beta_1(i)}{\sum_{i=1}^{N} \alpha_t(i)\beta_t(i)} \tag{44}$$

$$a_{ij} = \frac{\sum_{t=1}^{N} \xi_t(i,j)}{\sum_{t=1}^{T-1} \gamma_t(i)} = \frac{\sum_{t=1}^{T-1} \alpha_t(i)a_{ij}b_j(O_{t+1})\beta_{t+1}(j)}{\sum_{t=1}^{T-1} \alpha_t(i)\beta_t(i)} \tag{45}$$

$$b_{jk} = \frac{\sum_{t=1}^{T} \gamma_t(i)\chi(O_t)}{\sum_{t=1}^{T} \gamma_t(j)} \tag{46}$$

where the confidence factor for true statements $\chi[.]$ is 1, otherwise it is 0.

3.1.8. Viterbi Algorithm

The fundamental component of modern information technology is the Viterbi algorithm. The Viterbi algorithm implements the "maximum likelihood" Machine Learning decision rule. Using this algorithm, with a given model λ, the optimal state sequence q* is determined from the a- posteriori probability P(q|O, λ) [25]. This means that Eq. (47) represents the a-posteriori probability;

$$P(q|O, \lambda) = \frac{P(O, q \vee \lambda)}{P(O \vee \lambda)} \tag{47}$$

and q* is an optimal state set if the following assumption is valid as in Eq. (48):

$$P(O, q^*|\lambda) = max_{q \in Q^T} P(O, q|\lambda) = P^*(O \vee \lambda) \tag{48}$$

In other words, the Viterbi-algorithm calculates the probabilities (as in Eq. (49)) as:

$$\vartheta_t(j) = max_{q \in Q^T}\{P(O_1 \dots O_t, q_1 \dots q_{t-1} \vee \lambda) \vee q_t = j\} \tag{49}$$

Like the forward or backward algorithm, the Viterbi probabilities can be calculated as follows using the Eqs. (50), (51), (52), (53) and (54).

- **Initialization**

For all j=1, …, N put $\vartheta_1(j) = \pi_j b_j(O_1), \psi_1(j) = 0$ (50)

The vector $\psi_1(j)$ denotes the predecessor of state S$_i$ in the path to $\vartheta_1(j)$

- **Recursion**

For t>1 and all j=1, …, N put

$\vartheta_t(j) = max_j\left(\vartheta_{t-1}(i)a_{ij}\right)b_j(O_t)$ (51)

$\psi_t(j) = argmax_i\vartheta_{t-1}(i)a_{ij}$ (52)

- **Termination**

calculating

$P^*(O|\lambda) = max_j\vartheta_T(j)q *_T= argmax_j\vartheta_T(j)$ (53)

- **Traceability**

Reconstruct an optimal sequence using Eq. (54).

$q *_t= \psi_{t+1}q *_{t+1}\ t = T-1, \ldots ,1$ (54)

Since the direct re-estimation of the probabilities of Eq. (51) results in the numbers becoming too small (under-estimation), the logarithmic probabilities are calculated as in Eq. (55):

$\vartheta_t(j) = max_j\{\left(\vartheta_{t-1}(i) + log\left(a_{ij}\right)\right\} + logb_j(O_t)$ (55)

The function of the Viterbi algorithm can be described with a trellis diagram shown in Fig. 6. As depicted in the figure, the possible states (1 to 3 in this case) are plotted against the time index of the observation. The nodes that lie on top of each other symbolize the possible states per time step with their symbol probabilities at any point in time. The connecting lines indicate the transition probabilities to the states of the next point in time. Using this trellis diagram, the best path to one of the states can be found iteratively according to the observation by adding the logarithmic transition probabilities to the logarithmic symbol probabilities. The maxima found then represents the most probable sequence of states.

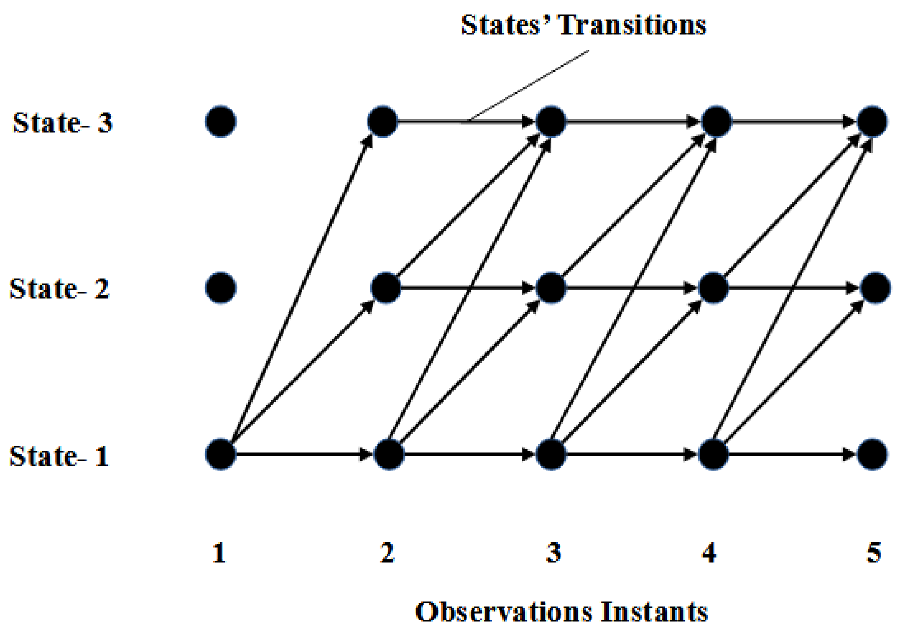
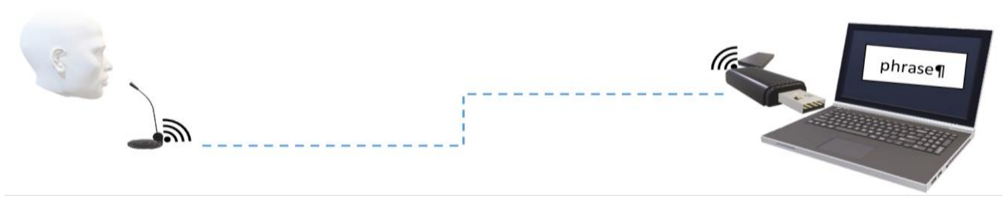


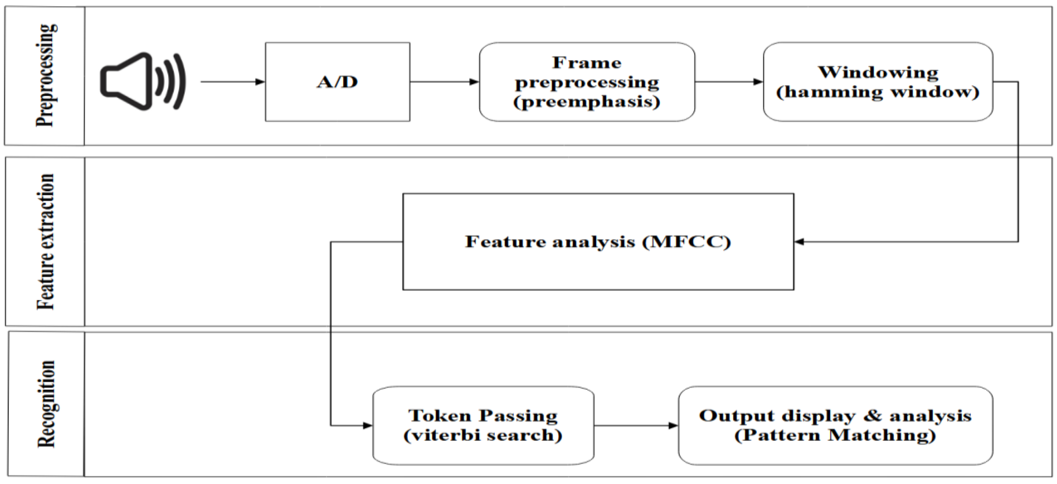
Fig. 6. The Viterbi trellis diagram.

3.2. System Architecture

As shown in Fig. 7, the designed prototype has mainly three modules i.e., signal preprocessing, feature extraction and recognition modules. The following subsection describes details of the implemented system.



(a)


Preprocessing
A/D
Frame preprocessing (preemphasis)
Windowing (hamming window)
Feature extraction
Feature analysis (MFCC)
Recognition
Token Passing (viterbi search)
Output display & analysis (Pattern Matching)

(b)

Fig. 7. a) System architecture overview; b) system function modules.

3.2.1. Data Preparation

The first step in realizing a speech recognizer with the HTK is the data preparation, which includes the following preparations: Grammar definition: First, a so-called grammar is defined. In this work, the grammar consists of ten numbers (zero to nine) and ten command words. A dictionary with the associated phonemes is then generated, whereby the words must first be sorted to adapt them to British English Pronouncing Dictionary (BEEP).

If no language data is available, new ones can be recorded and labeled with the HTK tool HSIab. This work used speech data from the American National Institute for Standards and Technology (NIST) speech databases. The voice data is header less PCM data (NOHEAD). They consist of 8 female and 8 male voices recorded at 12.5 kHz (PCM) and each labeled for training and testing. Transcription data: the labeled word data is combined into a script (MLF, master label file). Depending on the level of training, word data can be edited from the MLF using the HTK label editor as HLed phoneme data. Data encoding: the final step in data preparation is data encoding. This is done with the HTK tool HCopy, with the data possibly being parameterized as MFCC or LPC. The parameter format used in this work is MFCC. A script and a configuration file are required for coding.

3.2.2. Data Modeling

After the data has been prepared, the model is created, which includes the following steps: Creation of monophonic sounds; First, an initial monophone must be created. In this work, the "flat stall" monophone was generated using HCompV. This module computes the global mean and covariance of the training data, which is then used to initialize the parameters of the HMMs, thereby equating all mean and covariance components. HCompV calculates a vector equal to 0.01 of the global variances (also called "variance floor macros"). The values of this vector are the limits of the variance of the estimate. For this purpose, a left-right HMM prototype with eight states and initially two mixtures are defined.

To make the system more robust against background noise, a pause model is introduced and the HMM structures are edited with the HTK Tool (HHEd). The silence model here is a left-right HMM with 3 states. In contrast, the pause model has only one state. The emissive state of the pause model is connected to the central state of the silence model, as shown in Fig. 8.

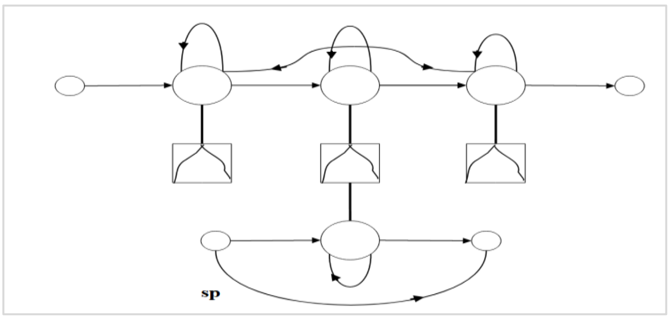


Fig. 8. The Tee-model.

The next step of modeling is data alignment to consider different ambiguous phoneme combinations in the dictionary. This is done by first executing the HTK detection program package HVite once and then executing the HERest. The final step in modeling is the generation of contextual triphones from the previous monophones. For this purpose, the monophones are cloned with triphone transcription using the HTK tool (HLed) and a re-estimation is carried out.

3.2.3. Model Training

The word training describes the determination of parameters that determine the behavior of the speech recognizer. HMM can be trained at different levels. Depending on whether a phoneme model or a single-word model is sought, an HMM can appear on the monophonic level, the context dependent triphone-level or the word levels.

The HTK tools, HCompV and HERest are used for the monophones and context-dependent triphone training described above. However, the word level training is carried out with Hlnit and HRest, whereby the method remains the same. An HMM has multiple Gaussian densities, which can take a long time to decode. However, this is not desirable for real-time detection. For this reason, an attempt was made in this work to reduce the computing effort of the acoustic HMM for the PC and later for the DSP. The single-word models were trained at the word level, which means that each word is modeled by a single HMM. This procedure has the advantage that recognition is improved because the word-dependent variability is also modeled. But it has the following disadvantages: more extensive speech data is required for the training. The system is limited in its usefulness by well-defined and small vocabulary applications.

3.2.4. Model Testing

After the training, the speech recognizer was tested. A test was carried out on each of the 3 levels mentioned above. HVite, as mentioned earlier, is the HTK detection suite. This module is an implementation of the Viterbi algorithm described above. For recognition, HVite reads the parameters of the command line and then makes the appropriate settings. The input buffer is first opened and filled with parameterized language data. The parameterization or calculation of the feature vectors is done with the modules HWave and HParm. HVite then use the token-passing algorithm to find the path with the highest logarithmic probability through the HMM network.

Upon detection, HVite creates a lattice containing the best state sequence. This Lattice can be saved to a file as a Standard Lattice File (SLF) and use again for recognition later. The lists of the best word hypotheses are generated using this lattice and stored in a transcription file, taking only the best hypothesis.

3.2.5. Model Evaluation

The HTK toolkit also includes a program package HResults for evaluating a speech recognizer. HResults reads the HVite output and compares it to a reference transcription file. The evaluation is based on the following two aspects: the accuracy in terms of the number of correctly recognized words and if there are errors, it will indicate whether it is a word deletion (deletion), insertion (insertion) or interchange (substitution). The number of correctly recognized words and the accuracy are given by:

$$\%correct = \frac{H}{N} x100\%$$

$$Accuracy = \frac{H-I}{N} x100\%$$

where H = number of correct labels, I = number of insertions and N = total number of labels in the transcription file. Fig. 9 depicts graphically the model evaluation process.

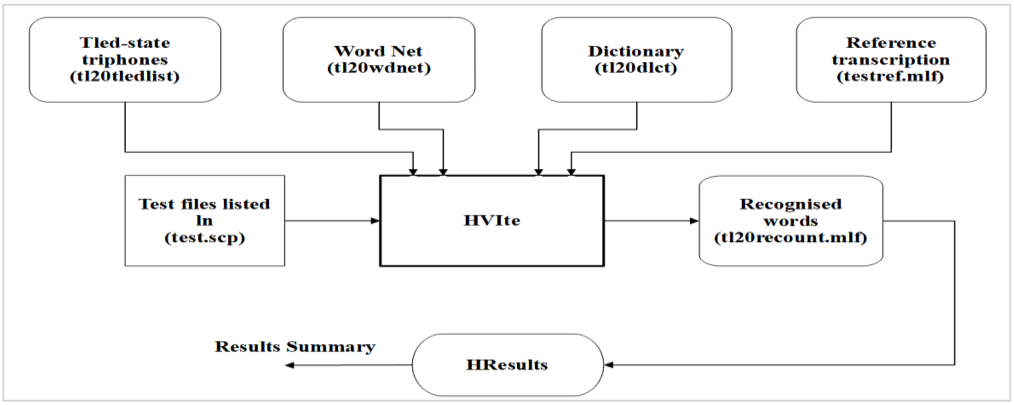

Tled-state triphones (tl20tledlist)
Word Net (tl20wdnet)
Dictionary (tl20dlct)
Reference transcription (testref.mlf)
Test files listed In (test.scp)
HVIte
Recognised words (tl20recount.mlf)
Results Summary
HResults

Fig. 9. The model evaluation process.

3.3. System Embedded Hardware Prototype

3.3.1. Embedded Prototype Structure

The implemented hardware prototype of the single-word speech recognition system consists of a TI evaluation board equipped with a TMS320C6701 digital signal processor, a PC, a JTAG emulator, an A/D converter and a microphone as shown in Fig. 7(a) [26]. Signal processing and control take place via software and explained in the following subsections.

The main building block of the EVM board is the DSP. This is a floating-point DSP with a so-called VelocTi VLIW architecture. VelocTi is a deterministic architecture, with fewer restrictions on how and when instructions are executed or loaded. Accordingly, the program and data buses are separated. The CPU has eight independent units consisting of two multipliers and six arithmetic logic units (ALUs). The CPU has 32 32-bit registers and can be addressed with 8, 16 or 32 bits. It supports both "big-endian" and "little-endian" mode addressing and can execute up to eight 32-bit commands per second. There are two data paths (data path A and B) for the CPU for data processing. Each data path has four instruction units (.L, .S, M, .D) and a register file with six 32-bit registers. The instruction units perform the logic, shift, multiply, and data address operations. Except for load and store instructions, all instructions are performed in the registers. For all data exchange between the register file and memory, the data addressing units (.DI, D2) are stated alone. The registers on the CPU only support one read or write instruction per instruction unit (cycle). There are two 32-bit paths for loading data from memory to register files; These are LDI for register file A and LD2 for register file B. The C6701 processor also has two 32-bit data paths (.STI, .ST2) for loading the register values from each register (A, B) to memory. The C6701 has 64KB internal program memory (IPRAM) and 64KB internal data memory (DRAM), whereby the IPRAM can also be used as a program cache. The IDRAM consists of two blocks with eight 16-bit banks, which enable parallel and double precision loading of the CPU. In addition to the two internal memories, using the EMIF (external memory interface) 256 KB synchronous SBSRAM (synchronous burst static random-access memory) and 2.x16 MB SDRAM (synchronous dynamic random-access memory) memory extensions supported, there are four DMA (direct memory access) channels for direct memory access. This, in combination with the HPI (host port interface), allows access to the entire memory space. The two serial ports (McBSPs, multi-channel buffered serial ports) ensure the serial arrangement of an audio, communication, or other device. Two clocks are available for timing, interrupts, and other purposes. The C6701 EVM Board can be operated at 160 MHz or 133 MHz.

3.3.2. Real Time Data Exchange

Here the analog speech signal is first sampled and then quantized (AD conversion). The speech signal is now amplified as a function of frequency, with the areas with higher frequency components being boosted more than areas with smaller frequency components (pre-emphasis). This is done using a one-coefficient digital filter, sometimes called a pre-emphasis filter. With the help of a Hamming window, the speech signal is segmented into short periods of 20 ms. Because the time slices are small, the speech segments can be considered stationary. The Hamming window describes a time interval by a weight function (see Eq. (6)) with which the sample values of the interval are multiplied. The values near the center of the window are weighted more heavily than the outer parts. The loss (spectral leakage) caused by this segmentation is kept as small as possible by overlapping the segments. That is, the steps from one window to the next are smaller than the length of the window intervals. As a rule, every 10 ms a window of 25 ms width is considered.

3.3.3. Feature Extraction

The goal of feature extraction is the conversion of the speech signal into one that is favorable for further modelling depiction. The Mel-frequency cepstral coefficient analysis is used here to extract the features from the windowed speech segments. The segmented speech signal is fed to a filter bank whose filters are roughly evenly distributed on the Mel scale. The filter coefficients obtained in this way are then logarithmized and then subjected to a discrete cosine transformation.

The parameters for the energy and deltas (derivatives) are calculated for the resulting coefficients, also known as Mel-Frequency Cepstral Coefficients (MFCC). In this work, 12 MFCCs and a parameter for the energy are calculated, whereby the resulting feature vector has 39 elements, i.e., 13 for the calculated parameters and 13 for first and second derivatives.

3.3.4. Embedded Recognition on the DSP

A Viterbi search is carried out here using the Viterbi algorithm. In this process, the DSP compares an unknown input word with the words stored in memory and selects the most similar word. This process is additionally explained in the software implementation section.

In contrast to the PC, the DSP is an embedded system, so that a system, in this case the single-word speech recognizer, should consist of a program (PC source code), initialization and memory management. In short:

system = program + initialization + memory management

A program consists of algorithms and data structures. These parts of the program are referred to as sections on the DSP side. Dividing the codes and data into different sections provides flexibility on the DSP side because it allows storing the code section in ROM (read only memory) and storing the variables section in RAM (random access memory).

A software system on the DSP, unlike the PC, needs to be initialized. This is also one of the differences between the host (PC program) and the embedded system (DSP program). The description of the DSP and other peripheral initializations is omitted here and reference to [27-31] is made. The final part of the embedded system is memory management. This includes, among other things, the specification of the memory requirements of the system, the memory description for the linker and the code and data section allocation for the linker. To be able to determine the memory requirement, the source code is divided into the following code and data sections: Code section: vector code, system initialization code and executable function code (file.c, main.c), etc. Data section: global or static variables, initial values for global and static variables, dynamic heap, and stack for local variables, etc. The Linker Command File combines similarly named sections of the object files generated by the compiler and creates a new section at the specified location in memory. The output of the linker is an executable file (file.out) on the DSP, like that on the PC (file.exe). Since the source code does not fit into the faster internal memory (on-chip RAM), the external memory was also accessed. Access to the external memory is made possible by EMIF. The EMIF contains the SDRAM controller that executes the necessary control logic such as CAS (column address strobe), RAS (row address strobe) and refresh. The EMIF was configured using the EVM Technical Reference and datasheets, with the 3 clock values (timing values) being programmed in the control register.

The trained Hidden Markov Models are in the IDRAM (.arg section). All executable files are in the SDRAMO (text section) because they don't fit in the IPRAM. 2MB of the SDRAMI were released for the dynamic memory allocation. During implementation, it was important to know what limitations the hardware has in terms of the system's computing and recognition time. This has been studied and discussed under the Test and Evaluation section. Although the C6701 is a floating-point DSP, there were problems reading the PC-trained models. This problem was solved by coding the models in binary. Graphical analytic models for DSP system structures are alternatives that can be employed in providing solution to the aforementioned problem [32]. Additionally, OFDM system with half-symbol-spaced receiver and channel acquisition error over multipath fading channels can provide enhanced information rate to achieve reliability in communication [33].

3.4. System Workflow

There are three essential steps of the implemented single word recognizer prototype as illustrated in Figs. 10 to 13. First, the user data is requested and stored in an input buffer. The data is fetched into an intermediate buffer for further processing and emptied at the end of the process. Ultimately, the word is recognized and issued using the Viterbi search algorithm.

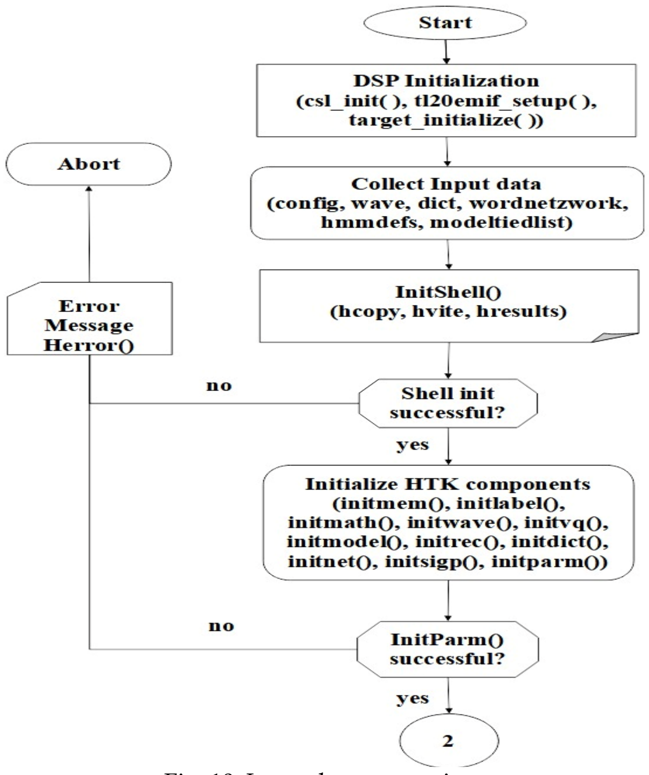

Start
DSP Initialization (csl_init( ), tl20emif_setup( ), target_initialize( ))
Abort
Collect Input data (config, wave, dict, wordnetzwork, hmmdefs, modeltiedlist)
InitShell() (hcopy, hvite, hresults)
Error Message Herror()
no
Shell init successful?
yes
Initialize HTK components (initmem(), initlabel(), initmath(), initwave(), initvq(), initmodel(), initrec(), initdict(), initnet(), initsigp(), initparm())
no
InitParm() successful?
yes
2

Fig. 10. Input data processing.

3.4.1. Input Data Processing

As shown in Fig. 10, the recognition process of the single-word speech recognizer begins with the initialization of the DSP and peripherals such as the external memory interface, the CSL (chip support library) and the RTDX channels. Thereafter, the dictionary, word network, and trained HMMs are loaded into RAM. Now the user data can be queried.

The input/output interface is initialized. If the initialization was not successful, an error message is output, and the program is aborted. Otherwise, all HTK Toolkit components are initialized, checking again whether the language data input interface was successfully initialized. After entering the user data, the parameter determination begins.

3.4.2. Parameter Determination

Before the parameters are determined, the correct user input is checked. If something is missing, a message is issued. Otherwise, the configuration parameters are used. The memory is prepared, and the user options and arguments are processed in order, giving an error message if any of the options are incorrect. The voice data is fetched from the input buffer and encoded as MFCC. The memory is emptied after coding, with only the parameterized speech data being temporarily stored in the intermediate buffer, which are then used as input data for word recognition as shown in Fig. 11.

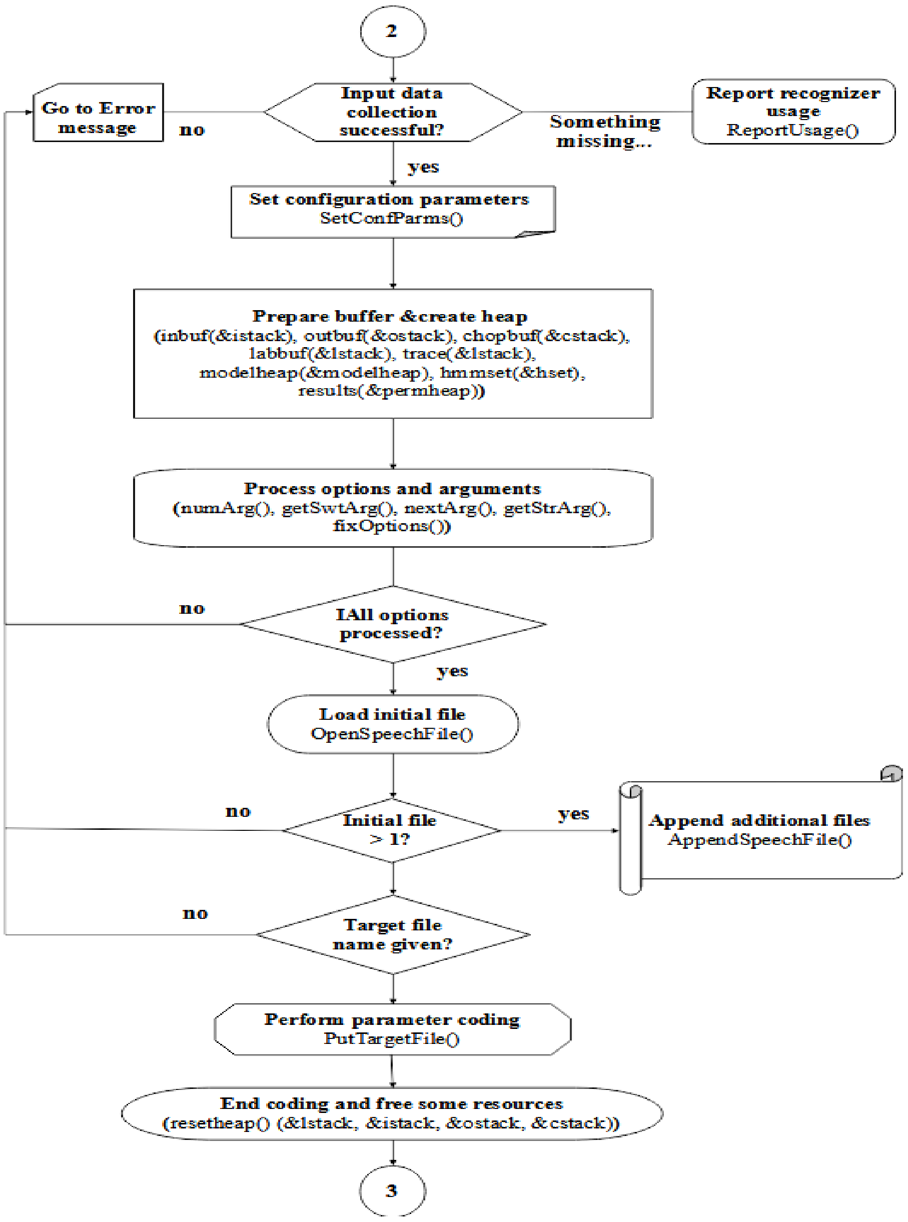

2
Input data collection successful?
Go to Error message
no
Something missing...
Report recognizer usage ReportUsage()
yes
Set configuration parameters SetConfParms()
Prepare buffer &create heap (inbuf(&istack), outbuf(&ostack), chopbuf(&cstack), labbuf(&lstack), trace(&lstack), modelheap(&modelheap), hmmset(&hset), results(&permheap))
Process options and arguments (numArg(), getSwtArg(), nextArg(), getStrArg(), fixOptions())
no
IAll options processed?
yes
Load initial file OpenSpeechFile()
no
Initial file > 1?
yes
Append additional files AppendSpeechFile()
no
Target file name given?
Perform parameter coding PutTargetFile()
End coding and free some resources (resetheap() (&lstack, &istack, &ostack, &cstack))
3

Fig. 11. Parameter determination.

3.4.3. Word Recognition and Output

The word recognition process begins by reading the models stored in RAM as illustrated with the flowchart in Fig. 12. A check is made as to whether the recognition process should take place using the specified detection network, if not; a so-called word alignment is performed. In this case, each word is compared with those in the dictionary.

In this work, a word loop was chosen as the word network, where any word can follow any other. Using the token-passing algorithm, a variant of the Viterbi algorithm, the DoRecognition function now finds the path with the highest logarithmic probability and thus the best path through the HMM network. If a word is recognized, it is saved in a MLF (master label file, temprecout.mlf) for later evaluation. The recognition process is ended and the RAMs that are no longer required are emptied.

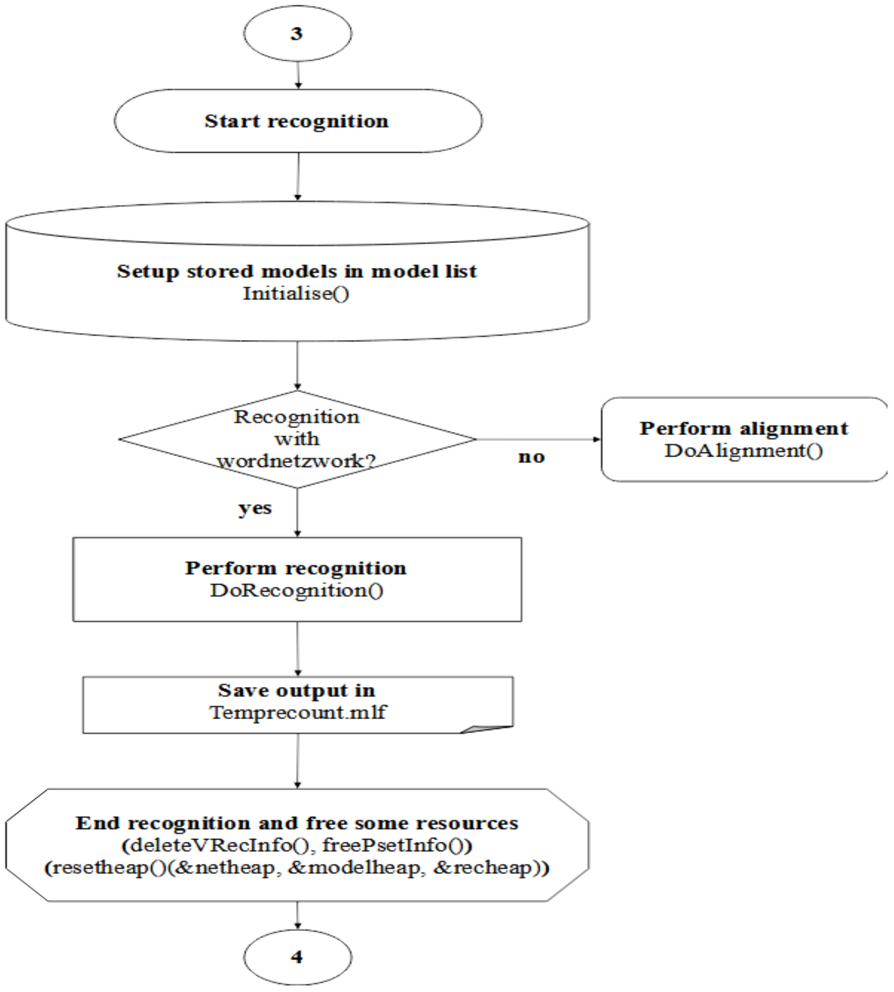

3
Start recognition
Setup stored models in model list
Initialise()
Recognition with wordnetzwork?
no
Perform alignment
DoAlignment()
yes
Perform recognition
DoRecognition()
Save output in
Temprecount.mlf
End recognition and free some resources
(deleteVRecInfo(), freePsetInfo())
(resetheap()(&netheap, &modelheap, &recheap))
4

Fig. 12. Word recognition and output.

4. RESULTS AND DISCUSSION

As shown in Fig. 13, the output file from the recognition process is read in to determine the recognition rate. For this purpose, a reference transcription file stored in the buffer is read. The pattern matching process begins and the counter is incremented and checked whether the counter has counted accurately. If it has more or fewer words than available, an error message is issued. Otherwise, a detection statistic is output.

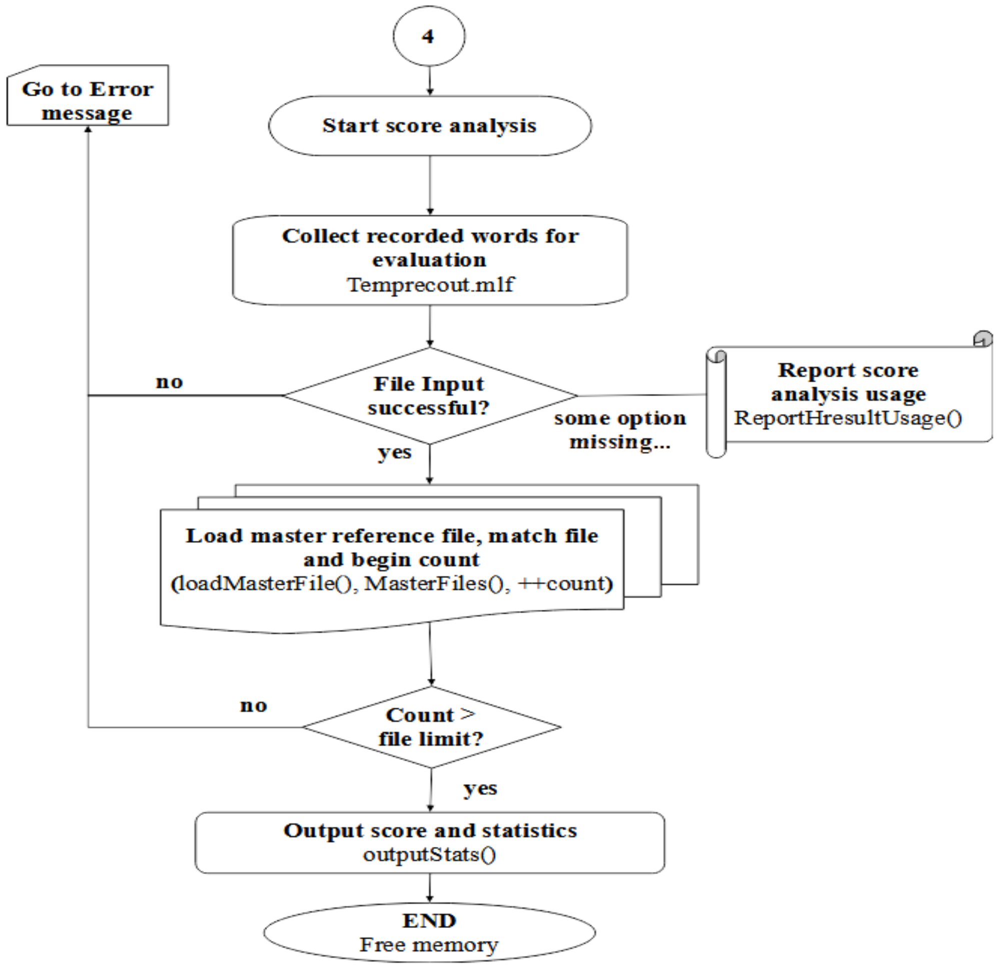

4
Start score analysis
Go to Error message
Collect recorded words for evaluation
Temprecout.mlf
no
File Input successful?
yes
some option missing...
Report score analysis usage
ReportHresultUsage()
Load master reference file, match file and begin count
(loadMasterFile(), MasterFiles(), ++count)
no
Count > file limit?
yes
Output score and statistics
outputStats()
END
Free memory

Fig. 13. Test process flow chart.

As already mentioned, the recognition rate of a speech recognition system depends on many factors. If the models are well trained, the recognition rate increases. Good training depends, among other things, on the language data used. The language database used is of crucial importance in training and evaluation. A language database (T146) from the National Institute of Standards and Technology (NIST) of the US Department of Commerce was available for both training and testing of this work. This is 12.5 kHz PCM voice data with a total of 16 speakers (8 women and 8 men), each speaking 46 different words. The language database is divided into a-training and a-test sections, which were also used as such for testing reliability.

The speakers of both sections were divided into 8 regions according to the geographic divisions of the United States of America, whereby one region is reserved for speakers who, due to frequent moves, could not be assigned to a geographically bound region. It follows from this information that the language variability discussed in the first section is contained in the language data. During the construction of the HMMs, problems arise when selecting a suitable base model and optimizing the parameters of the HMM. In search of a well-trained model, different MFCC levels (mfcc-0, mfcc-0DA, mfcc-EDA) were derived and then the mixtures were increased. Since this is a single-word speech recognizer with a language range of twenty words, it is relatively easy to form an HMM for each word. The following tables summarize the results. To be able to determine how well trained the HMMs are, the evaluation was carried out on the PC. An HMM with different states was chosen as the basic model. Table 1 contains recognitions test results for different mixtures.

Table 1. Results of recognitions test for different mixtures.

Parameter	Mixture			
	3	5	7	8
MFCC_0	48.18	54.13	57.12	69.76
MFCC_EDA	54.24	61.23	67.25	72.67
MFCC_0DA	57.21	60.11	59.00	74.80

As observed in Table 1, the MFCC_EDA had the most recognition in the 5th and 7th mixture while the MFCC_ODA recorded the most in the 3rd and 8th mixture. Hence, an individual assessment of the various contributing parameters is known and taken into consideration.

Graphs in Figs. 14 and 15 show the results of the various parameters. From the figures, it is clear that the choice of MFCC parameters influenced the recognition rate.

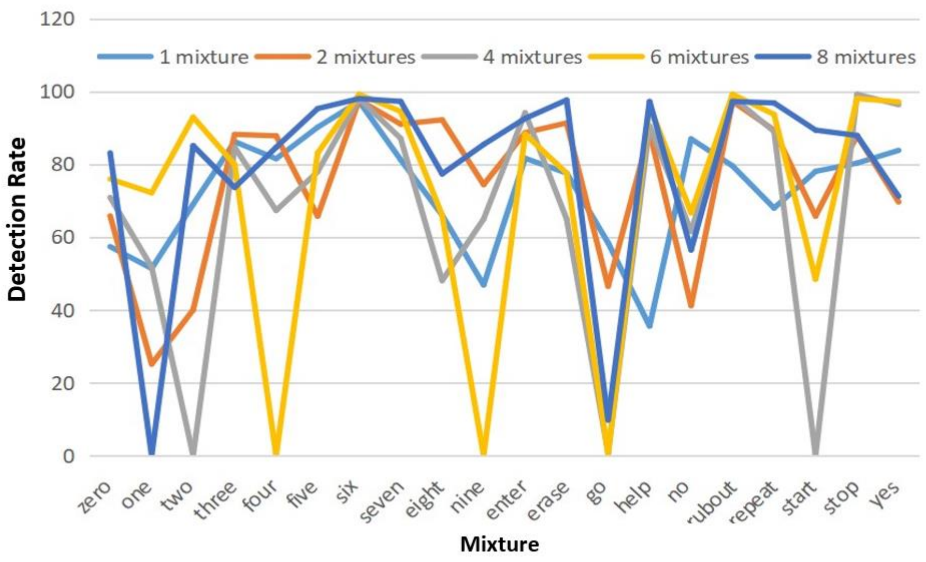


Fig. 14. The detection rate of several mixtures (20 iterations, 8 states).

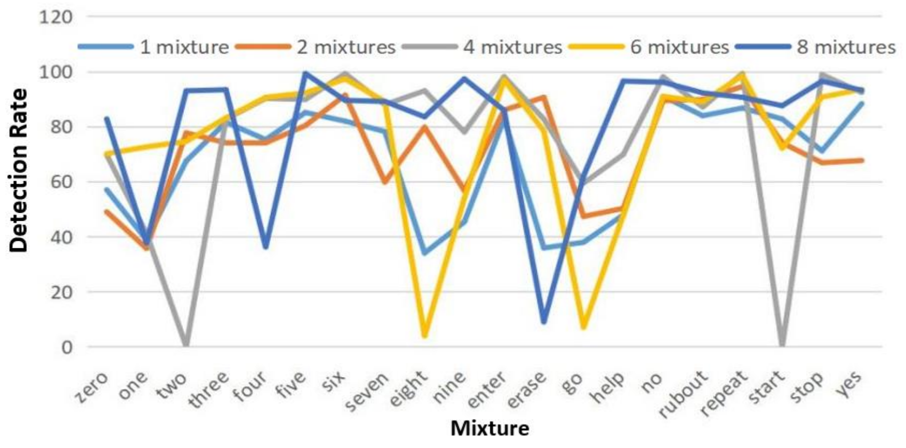


Fig. 15. The detection rate of several mixtures (20 iterations, 7 states).

To increase the mixtures, HMM prototypes of different mixture levels were created. With the help of these prototypes, the models were retrained and the influence on the individual words and the entire system is evaluated. From the graphs, it can be seen that increasing the number of mixtures increases the detection rate, but not automatically. Since the test material came from NIST, the evaluations were carried out using the NIST scoring format. Word error rate means the number of word errors divided by the number of words in the reference test file. There are three types of word errors: 1) Substitution error: this error occurs when a word difference is found when comparing the reference file to the system output file; 2) Word deletion (deletion error): this error occurs when no corresponding word can be found in the system output file for the referenced word; and 3) Word insertion (insertion error): the error occurs when no corresponding word can be found in the reference file for the system word.

The reference test file consists of word sequences which can be assumed to be the best-spoken words for the corresponding speaker. Table 2 shows test results of a single word recognized line in a corresponding NIST format.

Table 2. Recognition rates in NIST scoring format.

Total number of words	14960	%
Total word errors	3556	21.10
With substitutions	3556	21.10
With deletions	0	0
With insertions	0	0
Correct words	11804	78.90

After determining the recognition rate with the available NIST test data, it was also tested with other speech data. The recognition time of the speech recognizer was about two seconds with the PC, with the Simulator about six minutes and with the DSP about one minute. To make it easier to improve subsequent versions, the execution time of the functions in the system was determined. The following tables show these parameters. Table 3 contains results of the execution times by various parameters and Table 4 shows other executions.

Table 3. Execution times using different parameters.

Algorithm	Execution time / cycles
Hamming Window	649686
Pre-emphasis	589893
FFT	1887011
Cepstrum Weighting	32108
Log Energy Normalization	8052
Cepstral Window Generation	4176
Wave to MFCC Analysis	40287657
Mel Scaling	487306
Frame Conversion	43664639
Add Derivatives	641690
Filter Bank Analysis	584712

Table 4. Results of other execution times.

Task	Algorithm
Use of Configurations parameter	Execution time per cycles
Inputting the Word Networks	6261
Observation processing	3270311
Average values calculation	25493596
Variance calculation	335606
Mixture calculation	372144
Reading the HMM definition	1757556
Loading the HMMs	1681180
Dynamic Memory Space Location	3891153
Reading words from dictionary	2363761
Word recognition in lattice data storage	145637
Wave data collection for processing	7003
Reading configuration data	715863
Using configuration data	766257
Options processing	6261
Reading of frames in wave data	419485

5. CONCLUSIONS

Interpersonal communication does not take place in spoken language alone; it is supported by non-verbal means such as facial expressions and gestures. In the context of this work, speech recognition was considered; its description, feature extraction, modeling, and recognition. The temporal resolution of the speech signal in a sequence of linguistic units (words and phonemes) can be done using dynamic time warping (DTW). This can be used in the same way with Hidden Markov Model (HMM), since these are memory-free. This paper focused on the description of the HMM and the actual algorithms such as the forward-backward algorithm, the Baum-Welch algorithm, and the Viterbi algorithm, which were also used in the software implementation. The philosophy and design of the HTK Toolkit has been described. This work shows how a speech recognizer can be constructed from the forward-backward algorithm and the expectation-maximization decision rule. Using the HMMs, both the speech model and the acoustic model of the speech recognizer could be formulated mathematically. The Viterbi algorithm can be used for a focused search to determine the best path through the word network and thus the optimal word order. The speech recognizer was tested on the PC and on the TI C6701 simulator and EVM board. Several MFCC parameters were chosen for testing purposes. Script files were written and used in conjunction with the HTK Toolkit. Experiments were carried out and by varying and combining these parameters, the impact on the recognition rate was examined. The models were trained individually, with the dictionary also being written manually and C programs written for testing purposes. Functions and algorithms such as the Viterbi, Baum-Welch and forward-backward algorithms were borrowed from the HTK Toolkit, and the system was adapted accordingly. The complete single word recognition system was tested with different speech data formats and the results were evaluated. Finally, the code was implemented, tested, and made real-time capable on the TI TMS320C6701. A so-called linker command file was written, to ensure that the constants, variables, data, and executable functions in the source code are written to the appropriate locations in memory. The system was made suitable for real-time data exchange using DSP/BIOS and RTDX channels. Hence, the approximately 79% correct recognition results recorded proved that this system can be used to aid dictation of speech impaired persons or applications in real time with a 10 ms data exchange rate.

Authors would like to emphasize the need to improve the speech recognition rate to higher percentage values by reducing the percentage of the total word error to single digit value in future works.

Acknowledgement: The authors would like to appreciate and acknowledge Professors Reinhold Orglmeister and Dorothea Kolossa formerly of the Technical University of Berlin for their supervision of the initial project.

REFERENCES

[1] P. Mustillo, *Human-Computer Interaction*, Nortel, COMP 675.

[2] S. Young, The HTK Hidden Markov Model Toolkit, 1994. <https://htk.eng.cam.ac.uk/ >

[3] S. Young, G. Evermann, D. Kershaw, G. Moore, *The HTK Book*, Microsoft, ver.3.1, 2001.

[4] H. Toylan, E. Türkeş, E. Çağlarer, "Real-time control of mobile robot using HMM-based speech recognition system," *Anadolu University Journal of Science and Technology A- Applied Sciences and Engineering*, vol. 18, no. 5, pp. 897-907, 2017.

[5] A. Benmachiche, A. Makhlouf, "Optimization of hidden markov model with gaussian mixture densities for arabic speech recognition," *WSEAS Transactions on Signal Processing*, vol. 15, pp. 85-94, 2019.

[6] H. Shu, I. Hetherington, J. Glass, "Baum-Welch training for segement-based speech recognition," *IEEE Automatic Speech Recognition and Understanding*, pp. 43-48, 2003.

[7] A. Sosiawan, R. Nooraeni, L. Sari, "Implementation of using HMM-GA in time series data," *Procedia Computer Science*, vol. 179, pp. 713-720, 2021.

[8] M. Hanif, F. Sami, M. Hyder, M. Iqbal, "Hidden markov model for time series prediction," *Journal of Asian Scientific Research*, vol. 7, no. 5, pp. 196-205, 2017.

[9] F. Shi, X. Cheng, X. Chen, "The summarize of improved HMM model," *Proceedings of the 2012 2nd International Conference on Computer and Information Application*, pp. 0627-0630, 2012.

[10] N. Pai, J. Chen, P. Chen, J. Hong, "Speech processing based on hidden markov model and vector quantization techniques applied to internet of vehicles," *Sensors and Materials*, vol. 30, no. 4, pp. 803-820, 2018.

[11] B. Sonkamble, D. Doye, "Hidden markov model for speech recognition using modified forward-backward re-estimation algorithm," *International Journal of Computer Science Issues*, vol. 9, no. 2, pp. 242-247, 2012.

[12] M. Pietras, P. Klęsk, "FPGA implementation of logarithmic version of baum-welch and viterbi algorithms for reduced precision hidden markov models," *Bulletin of the Polish Academy of Sciences, Tennical Sciences*, vol. 65, no. 6, pp. 935-947, 2017.

[13] J. Lember, A. Koloydenko, "The adjusted viterbi training for hidden markov models," *Bernoulli*, vol. 14, no. 1, pp. 180-206, 2008.

[14] I. Sassi, S. Anter, A. Bekkhoucha, "A new improved baum-welch algorithm for unsupervised learning for continuous-time HMM using spark," *International Journal of Intelligent Engineering and Systems*, vol. 13, no. 1, pp. 214-226, 2020.

[15] A. Churbanov, S. Winters-Hilt, "Implementing EM and viterbi algorithms for hidden markov model in linear memory," *BMC Bioinformatics*, vol. 9, no. 1, pp. 1-15, 2008.

[16] V. Tabar, D. Plewczynski, H. Fathipour, "Generalized baum-welch and viterbi algorithms based on the direct dependency among observations," *Journal of the Iranian Statistical Society*, vol. 17, no. 2, pp. 205-225, 2018.

[17] P. Bansal, A. Kant, S. Kumar, A. Sharda, S. Gupta, "Improved hybrid model of HMM/GMM for speech recognition," *International Book Series: Information Science and Computing*, pp. 69-74, 2008.

[18] R. Kanchan, M. Soni, "A research paper on isolated spoken word recognition using hidden markov model," *International Journal of Advance Research and Innovative Ideas in Education*, vol. 2, no. 4, pp. 101-110, 2016.

[19] C. Becchetti, L. Ricotti, *Speech Recognition: Theory and C++ Implementation*, First Edition, Chichester, John Wiley and Sons, 2000.

[20] P. Noll, *Nachrichtenübertragung I*, Institut für Telekommunikations syteme, TU Berlin, 1998.

[21] L. Rabiner, B. Juang, *Fundamentals of Speech Recognition*, Englewood Cliffs, New Jersey 07632, Prentice Hall, 1993.

[22] D. Kershaw, *Phone Context-Dependency in a Hybrid ANN/HMM Speech Recognition System*, M.S. Thesis, St. John College, University of Cambridge, 1997.

[23] U. Kiencke, *Ereignisdiskrete Systeme*, R. Oldenbourg Verlag, 1997.

[24] L. Rabiner, "A tutorial on hidden markov models and selected applications in speech recognition," *Proceeding of the IEEE*, vol. 77, no. 2, 1989.

[25] E. Günter, S. Talamazzini, *Automatische Spracherkennung: Statistische Verfahren der Musteranalyse*, Vieweg Verlag, 1995.

[26] Texas Instruments, *TMS320C6701 Evaluation Module Technical Reference*, TI Application Report, SPRU305, 1998.

[27] Texas Instruments, *TMS320C6000 Peripherals Reference Guide*, Texas Instruments, SPRUI190c, 2001.

[28] Texas Instruments, *TMS320C6000 DSP/BIOS Users Guide*, Texas Instruments, SPRU303A, 2001.

[29] Texas Instruments, *TMS320C62x/C67x CPU and Instruction Set*, Texas Instruments, SPRUI189c, 2001.

[30] Texas Instruments, *TMS320C6000 Technical Brief*, Texas Instruments, SPRUI197D, 1999.

[31] L. Swee, "Implementing speech recognition algorithms on the TMS320C2xx platform," *Texas Instruments, Application Notes*, 1998.

[32] M. Al-suod, A. Ushkarenka, A. Soliman, M. Zeidan, A. Awwad, A. Al-Quteimat, "Development of graphical analytical models for digital signal processing system structures," *Jordan Journal of Electrical Engineering*, vol. 6, no. 2, pp. 140-153, 2020.

[33] A. Alqatawneh, L. Al-Tarawneh, "OFDM System with half-symbol-spaced receiver and channel acquisition error over multipath fading channels," *Jordan Journal of Electrical Engineering*, vol. 7, no. 2, pp. 96–107, 2021.