CHAPTER 4

DESIGN OF TAMIL SPEECH QUERY RECOGNITION SYSTEM

Automatic Speech Recognition (ASR) decodes human voice to generate transcription using computer-based analysis. Several speech-based methods have been explored in IR community, which can be grouped into two categories, according to its application. They are listed below.

(i) Spoken Document Retrieval

(ii) Speech Drive (Spoken Query) Retrieval

The formal uses written queries, which are used to search speech databases for relevant information (Jones *et al.*, 1996; Sheridan *et al.*, 1997; Johnson *et al.*, 1999; Singhal and Pereira, 1999; Srinivasan and Petkovic, 2000; Wechsler *et al.*, 1998; Whittaker *et al.* 1999). Examples of speech databases include broadcast news audio, telephonic meeting circulars and Voice-based lessons for school and college students. The second group uses spoken queries to retrieve relevant textual information (Itou *et al.*, 2001; Kupiec *et al.*, 1994).

Initiated partially by the TREC-6 spoken document retrieval (SDR) track (Garofolo *et al.*, 1997), various methods have been proposed for spoken document retrieval. Speech driven text retrieval systems have huge advantages in several keyboards-less retrieval applications like telephone-based retrieval, car navigation systems and user-friendly interfaces. Inspite of these numerous applications, only a small number of methods have been proposed and explored for speech-driven text retrieval (Fujii *et al.*, 2002a).

Almost all of the available literature on speech query-based systems aim to improve the retrieval process and do not focus on improving the speech recognition part. The existing proposals use an existing speech recognition system with no enhancement or optimization operations. In other words, it

could be ascertained that the speech recognition and text retrieval tasks were treated as two independent modules that were connected as a way of input/output protocol.

Barnett *et al.* (1997) compared speech-driven retrieval with an existing text-based retrieval system called INQUERY using DRAGON speech recognition system. The test was based on 35 queries collected from TREC 101-135 topics, dictated by a single male speaker. Crestani (2000) also used the same input set and compared relevance feedback techniques with speech-driven text retrieval.

Speech recognition systems are domain and language specific and their performance with respect to recognition accuracy across domain are not satisfactory. The performance of the CLIR systems is directly proportional to the query input, which has to be short, meaningful and intelligently framed in order to retrieve documents that are more relevant to user request.

While using CLIR with a speech-driven system, it is imperative that the query text produced by the recognition engine is accurate, so that the CLIR produces relevant retrieval results. High error rate in speech query recognition will considerably decrease the performance of CLIR system. For example, the Barnett and Crestani systems showed an error rate of 30% was contributed due to the errors of speech recognition (Fujii *et al.,* 2002b). Moreover, almost all of the available solutions are used for English spoken queries recognition and text retrieval and only limited proposals have focused on Tamil speech query recognition whose results are used for text retrieval (Udhyakumar *et al.,* 2004; Plauche *et al.,* 2006; Barnard *et al.,* 2010; Godambe and Samudravijaya, 2011).

In order to solve these issues, this research work designs an enhanced ASR system for recognizing the spoken queries in Tamil language with the aim of improving both speech recognition and document retrieval tasks. The proposed ASR uses a neural network based approach for Tamil speech query recognition and has the features like being tolerant to noise, designed for Tamil

isolated word speech recognition, usage of neural network and speaker independent. The steps involved are listed below.

(i) Preprocessing

(ii) Feature Extraction

(iii) Speech Recognition

The proposed ASR is termed as ATSQR (Automatic Tamil Speech Query Recognition) System in this research work. The flow used by the proposed ASR system is presented in Figure 4.1. The following section presents details regarding each of these steps.

4.1. PREPROCESSING

The preprocessing step of ATSQR system perform two important tasks, namely,

(i) Pre-Emphasis

(ii) Noise removal

(iii) Windowing

(iv) Silence removal (Voice Activity Detection)

in order to enhance the quality of the speech data. This section presents the algorithms used to handle them.

4.1.1. Pre-Emphasis

In speech processing, the original signal usually has too much lower frequency energy, and processing the signal to emphasize higher frequency energy is necessary. Pre-emphasis, a very simple signal processing method, is used increase the amplitude of high frequency bands and decrease the amplitudes of lower bands. It is performed using Equation (4.1).

$$F_{Preem}(z) = 1 - a_{preem}(z^{-1}) \quad (4.1)$$

where F, a, z are ****. Pre-emphasis filter is one of the common filters used in speech enhancement which reduces background noise and resulting in good quality of speech. The original signal and their pre-emphasis of original signal for the word ‘amma’ are shown in Figure 4.2.

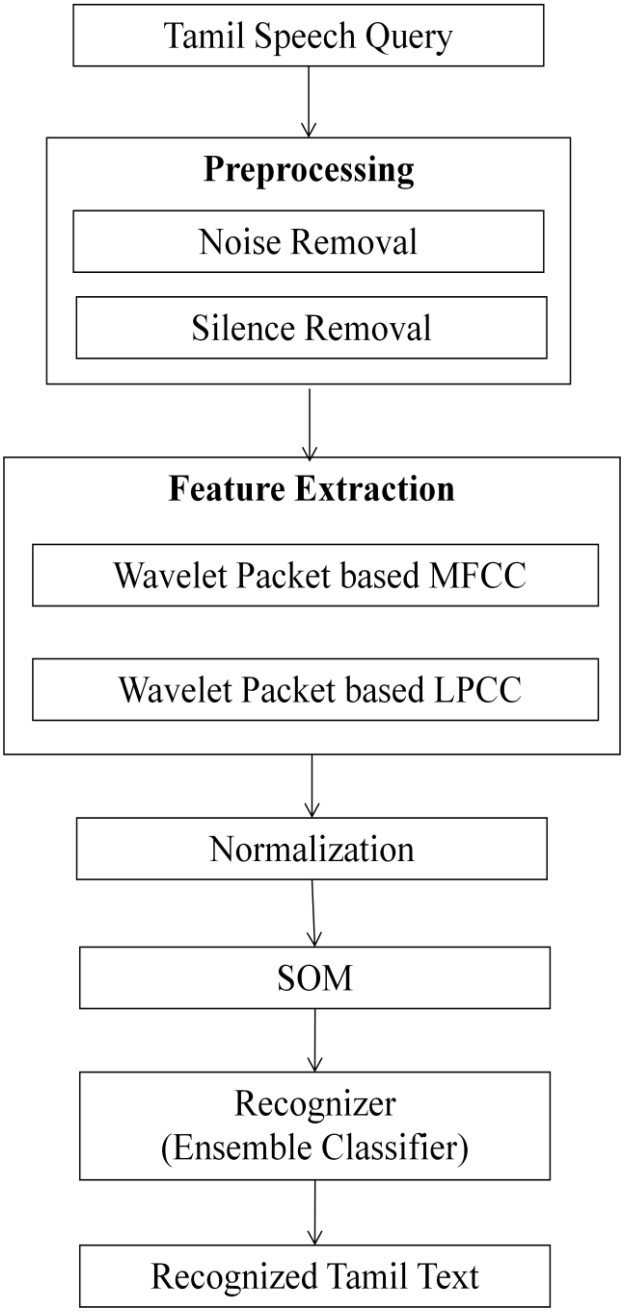


Figure 4.1 : Steps in Proposed ASR System

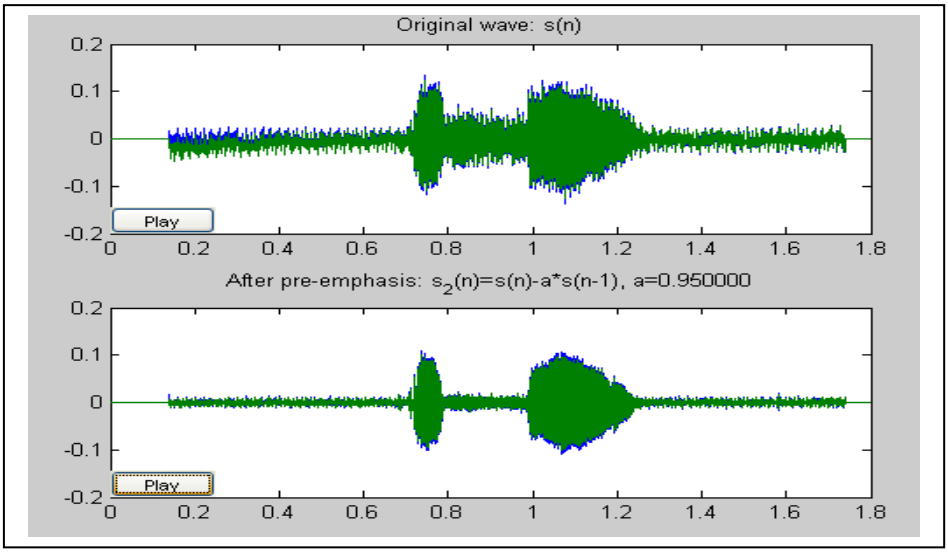


Figure 4.2 : Speech Waveform of Word "amma (mk;kh)" before and after pre-emphasis filter

4.1.2. Noise Removal Algorithm

Noise is ubiquitous in almost all acoustic environments. The speech signal that is recorded by a microphone is generally infected by noise originating from various sources. Such contamination can change the characteristics of the speech signals and degrade the speech quality and intelligibility, thereby causing significant harm to human-to-machine communication systems. Noise removal is a necessary step in TECLTR-S system as the presence of noise degrades the query quality and hinders in the performance of retrieval.

Noise removal or reduction in Tamil speech data is a challenging problem and many approaches have been proposed for this purpose. Examples include short-time spectral amplitude estimator (Jin *et al.*, 2014), signal subspace approach (Borowicz, 2015) and the human auditory system model-based approach (Maganti and Matassoni, 2014). Apart from these, several filters based on transformation techniques have also been used. Predominant among these are the use of are Fourier Transformation, Karhunen–Loeve transformation, cosine transformation, Hadamard transforms and wavelet transformation (Shrawankar and Thakre, 2010). This research work proposes a noise filter using Wavelet based on Switching Soft and Hard Thresholding.

This algorithm is termed as W2SHT algorithm in this research work and the steps involved are presented in Figure 4.3.

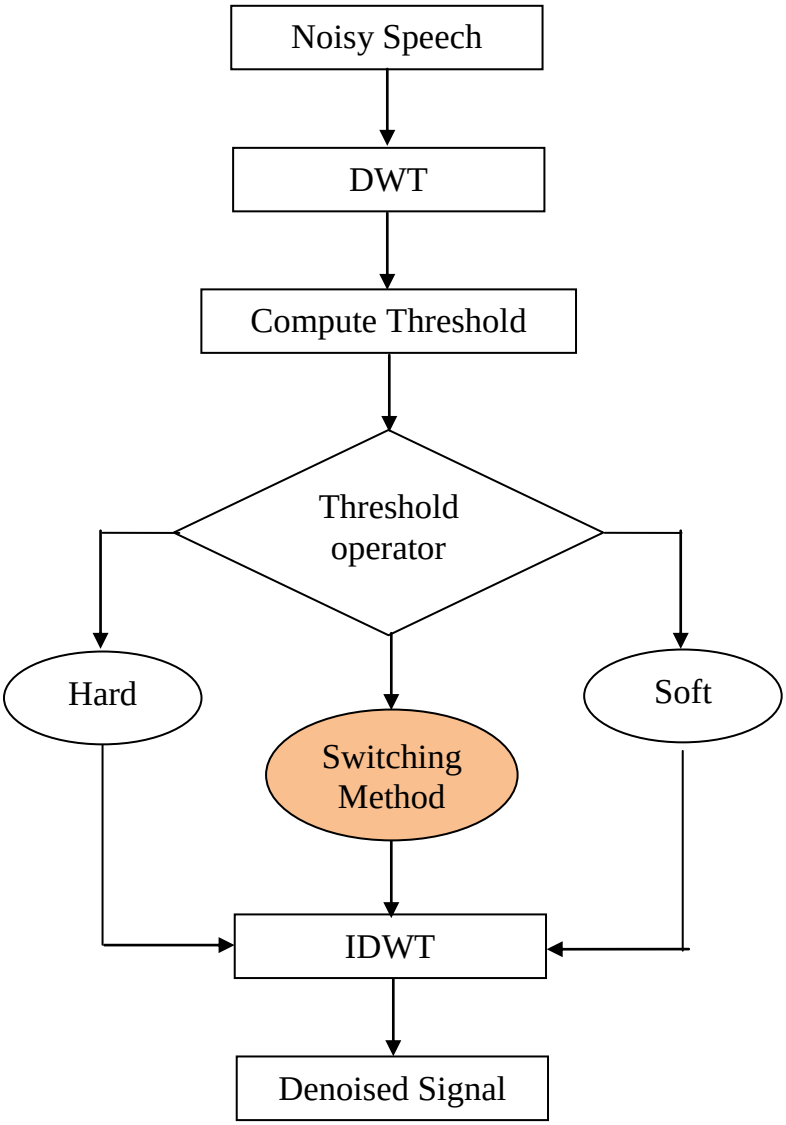


Figure 4.3 : W2SHT Algorithm

The W2SHT algorithm performs denoising in three steps after speech acquisition. The first step is a forward Discrete Wavelet Transform (DWT), followed by a thresholding step using switching hard and soft thresholding and the final step performs an Inverse Discrete Wavelet Transform (IDWT) to obtain the quality enhanced speech query.

- **Discrete Wavelet Transformation**

DWT-based algorithms have been used frequent in various fields of signal processing like image and speech. The reason behind this is the various advantages provided by these algorithms (https://www.clear.rice.edu/elec301/Projects01/dig_hear_aid/noisered.html). Wavelets are nonlinear functions and do not remove noise by low-pass filtering like many traditional methods. Low-pass filtering approaches, which are linear time invariant, can blur the sharp features in a signal and sometimes it is difficult to separate noise from the signal where their Fourier spectra overlap. For wavelets the amplitude, instead of the location of the Fourier spectra, differs from that of the noise. This allows for thresholding of the wavelet coefficients to remove the noise. If a signal has energy concentrated in a small number of wavelet coefficients, their values will be large in comparison to the noise that has its energy spread over a large number of coefficients. These localizing properties of the wavelet transform allow the filtering of noise from a signal to be very effective. While linear methods trade-off suppression of noise for broadening of the signal features, noise reduction using wavelets allows features in the original signal to remain sharp.

The DWT-based speech denoising algorithms use variable size time-windows for different frequency bands, thus resulting in high frequency resolution in low subbands (and low time-resolution) and low frequency-resolution in high bands. As a result, DWT is considered as a powerful tool for modeling non-stationary signals like speech, which exhibits slow temporal variations in low frequency and abrupt temporal changes in high frequency. It is more applicable for situations where speech is restricted to single-microphone speech enhancement (that is, only one noisy signal).

The DWT first performs decomposition of speech signal into coarse approximation and detail coefficients, which facilitate analysis at different frequency subbands at various resolutions. This is depicted in Figure 4.4.

Figure 4.4 : DWT Signal Analysis

Here, the wavelet transformation of the input signals x is estimated by passing it first through a low pass filter with impulse response 'g' resulting in a convolution of the two (Equation 4.2)

$$y[n] = (x * g)[n] = \sum_{k=-\infty}^{\infty} x[k]g[n-k] \tag{4.2}$$

At the same time, the signal is also decomposed using a high-pass filter h. The high-pass filter outputs detailed coefficients (high frequency signals) and low pass filter outputs approximation coefficients (low frequency signals). It is important that the two filters are related to each other and they are known as a quadrature mirror filter. The sub-signal produced from the low filter will have a highest frequency equal to half that of the original. According to Nyquist sampling, this change in frequency range means that only half of the original samples need to be kept in order to perfectly reconstruct the signal. More specifically, this means that, up-sampling can be used to remove every second sample. The approximation sub-signal can then be put through a filter bank and this is repeated until the required level of decomposition has been reached. This is represented as a binary tree with nodes representing a sub-space with different time-frequency localization. The tree is known as a filter bank. The sub sampling is performed using Equations (4.3) and (4.4).

$$y_{\text{high}}[n] = \sum_{k=-\infty}^{\infty} x[k]h[2n-k] \tag{4.3}$$

$$y_{\text{low}}[n] = \sum_{k=-\infty}^{\infty} x[k]g[2n-k] \tag{4.4}$$

Due to the decomposition process, the input signal must be a multiple of 2^n where n is the number of levels. In this research work a 4-level decomposition using Daubechies wavelet-4 (D4) algorithm is used.

The D4 algorithm belongs to the family of orthogonal wavelets defining a discrete wavelet transform and characterized by a maximal number of vanishing moments for some given support. With each wavelet type of this class, there is a scaling function (also called father wavelet) which generates an orthogonal multiresolution analysis. Daubechies D4 algorithm has four vanishing moments and encodes polynomials with two coefficients, i.e. constant and linear signal components.

The Daubechies D4 transform, as the name indicates, has four wavelet function coefficients (h_0, h_1, h_2 and h_3) and scaling function coefficients (g_0, g_1, g_2 and g_3). The scaling function coefficients are given in Equation (4.5) and the wavelet function coefficient values are given in Equation (4.6).

$$h_0 = \frac{1+\sqrt{3}}{4\sqrt{2}},\ h_1 = \frac{3+\sqrt{3}}{4\sqrt{2}},\ h_2 = \frac{3-\sqrt{3}}{4\sqrt{2}},\ h_3 = \frac{1-\sqrt{3}}{4\sqrt{2}} \qquad (4.5)$$

$$g_0 = h_3;\ g_1 = -h_2;\quad g_2 = h_1;\quad g_3 = -h_0 \qquad (4.6)$$

Each step of the wavelet transform applies the scaling function to the data input. If the original data set has N values, the scaling function will be applied in the wavelet transform step to calculate N/2 smoothed values. In the ordered wavelet transform the smoothed values are stored in the lower half of the N element input vector.

Each step of the wavelet transform applies the wavelet function to the input data. If the original data set has N values, the wavelet function will be applied to calculate N/2 differences (reflecting change in the data). In the ordered wavelet transform the wavelet values are stored in the upper half of the N element input vector. The scaling and wavelet functions are calculated by taking the inner product of the coefficients and four data values. The

Daubechies D4 scaling functions and wavelet functions are given in equations (4.7) and (4.8) respectively.

$$a_i = h_0 S_{2i} + h_1 S_{2i+1} + h_2 S_{2i+2} + h_3 S_{2i+3}$$
$$a[i] = h_0 S[2i] + h_1 S[2i+1] + h_2 S[2i+2] + h_3 S[2i+3] \quad (4.7)$$

$$c_i = g_0 S_{2i} + g_1 S_{2i+1} + g_2 S_{2i+2} + g_3 S_{2i+3}$$
$$c[i] = g_0 S[2i] + g_1 S[2i+1] + g_2 S[2i+2] + g_3 S[2i+3] \quad (4.8)$$

Each iteration in the wavelet transform step calculates a scaling function value and a wavelet function value. The scaling values, a_i, and the wavelet values, c_i are calculated by taking the inner product of the h_j and g_j coefficients and the signal. The index 'i' is incremented by two with each iteration, and new scaling and wavelet function values are calculated. In the above equations S_k ($1 \leq k \leq 2i$) values are signals and in the case of the forward transform, with a finite data set, i will be incremented until it is equal to N-2. In the last iteration the inner product will be calculated from s[N-2], s[N-1], s[N] and s[N+1].

- **Thresholding**

Wavelet thresholding technique (Bruce *et al.*, 1996; Donoho, 1992; Donoho and Johnstone, 1995) is a signal estimation technique that exploits the capabilities of wavelet transform for speech denoising. It removes noise by killing or shrinking coefficients that are insignificant relative to some threshold. The pseudo code for forward transformation is given in Figure 4.5.

They are simple yet effective and depend heavily on the thresholding parameter. The efficiency of WSD greatly depends on the correct choice of parameter. Wavelet thresholding is composed of two steps namely, thresholding method and threshold selection.

```
protected void transform( double a[], int n )
  {
    if (n >= 4)
    {
      int i, j;
      int half = n >> 1;
      double tmp[] = new double[n];
      i = 0;
      for (j = 0; j < n-3; j = j + 2)
      {
        tmp[i]      = a[j]*h0 + a[j+1]*h1 + a[j+2]*h2 + a[j+3]*h3;
        tmp[i+half] = a[j]*g0 + a[j+1]*g1 + a[j+2]*g2 + a[j+3]*g3;
        i++;
      }
      tmp[i]      = a[n-2]*h0 + a[n-1]*h1 + a[0]*h2 + a[1]*h3;
      tmp[i+half] = a[n-2]*g0 + a[n-1]*g1 + a[0]*g2 + a[1]*g3;

      for (i = 0; i < n; i++)
          {          a[i] = tmp[i];          }
    }
} // transform
```

Figure 4.5: D4 Forward Transformation

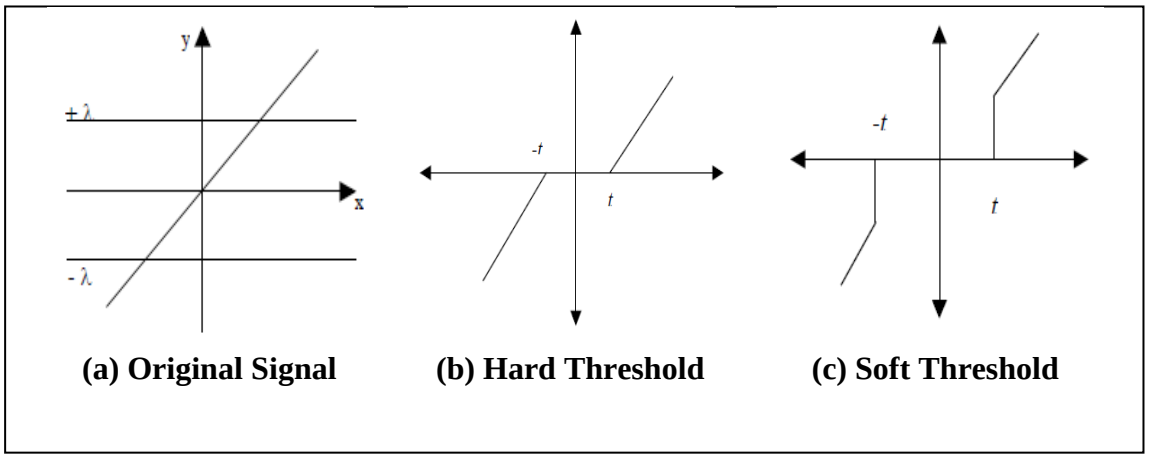
(i) Threshold operators

Most frequently used thresholding methods are soft and hard thresholding (Gabrea and Gargour, 2004). The hard and soft thresholding operations are defined as in Equations (4.9) and (4.10).

$$T_{hard}(I,\lambda) = \begin{cases} I & \text{for all } |I| > \lambda \\ 0 & \text{otherwise} \end{cases} \tag{4.9}$$

$$T_{soft}(I,\lambda) = \begin{cases} \text{sign}(I)\max(0,|I| - \lambda & \text{for all } |I| > \lambda \\ 0 & \text{otherwise} \end{cases} \tag{4.10}$$

The hard threshold work on the "Kill or Keep" principle where the input is kept, if it is greater than a defined threshold (λ) otherwise it is set to zero. It removes noise by thresholding only the detailed subband wavelet coefficients (I), while keeping the low-resolution coefficients unaltered. An extension to hard thresholding is the soft thresholding, which works on the "Shrink or Keep" principle. The output is forced to zero, if the absolute value of I is less than the threshold λ else the output is set to |I-λ|. The effect of hard and soft thresholding on an original signal is given in Figures 4.6a,b,c.



(a) Original Signal **(b) Hard Threshold** **(c) Soft Threshold**

Figure 4.6 : Effect of Thresholding

Discontinuities at ±λ is seen with hard thresholding and they are more sensitive to small changes, while soft threshold avoids both these situations. Thus the advantages of soft thresholding are it reduces abrupt sharp changes and provides an image whose quality is not degraded. Because of these advantages, soft thresholding is more frequently used. Soft thresholding can remove white and impulsive noises in the speech signal, but are not efficient in removing external noises like pen clicks, knocks on table, etc. In these situations, implementing hard thresholding has more advantages. But, as hard thresholding is not a best choice in removing noise, in this research work a hybrid method that switches between hard and soft thresholding is proposed.

The proposed switching method that combines the advantages of soft and hard thresholding method for removing noise in speech signal is given in Equation (4.11).

$$T_{switch}(I,\lambda) = \begin{cases} I * \dfrac{|I|^{\alpha} - \lambda}{|I|^{\alpha}} & \text{for all } |I| \geq \lambda \\ 0 & \text{otherwise} \end{cases} \qquad (4.11)$$

where I is the wavelet coefficient, λ is the threshold value and α is the parameter that performs the switching operation that controls the thresholding characteristics.

When the speech is dominated by inherent noise, soft thresholding is used and when it is degraded by the presence of external noise, hard thresholding is used. This can be controlled by selecting a suitable value for the parameter α in the above equation. The α parameter helps to estimate a better thresholding that takes into consideration the merits offered by both soft and hard thresholds. In this research work, the value of α was set to a range between 1 and 1.5 which was estimated using empirical experiments using speech data with varying noise standard deviation. The optimal value was estimated using Signal to Noise Ration (SNR) estimated between original degraded signal and denoised signal. When α is in this range, the proposed W2SHT algorithm performs a soft thresholding, else a hard thresholding is performed.

- **Threshold Selection**

Donoho and Johnstone (1995) derived a general optimal universal threshold under a mean square error (MSE) criterion. However, this threshold is not ideal for speech signals due to poor correlation between the MSE and subjective quality and the presence of correlated noise. Hence, an appropriate method to estimation the threshold, (λ), is required. While using wavelets, small coefficients across time and scale exhibit noise characteristics, while

large coefficients exhibit high energy signals. This fact can be used during the estimation of the threshold, λ. In normal practice, the threshold value is obtained by multiplying the selected threshold with the median value of the detailed coefficients at a particular level (Sumitra *et al.*, 2009; Sharma and Pyara, 2013).

In this method, the threshold (λ) is chosen according to the signal energy and the standard deviation (σ) of each wavelet subband. At each decomposition level, the σ of the noisy signal is calculated using Equation (4.12).

$$\sigma_j = \text{median}(|c_j|) / 0.6745 \quad (4.12)$$

where C_j are high frequency wavelet coefficients at jth level of decomposition and σj is Median Absolute Deviation (MAD) at this level. This standard deviation can be further used to set the threshold value based on the noise energy at that level. The modified threshold value (Johnson *et al.*, 2007) can be obtained by the Equation (4.13).

$$\lambda = \sigma_j \sqrt{2 \log(L_j \log_2 L_j)} \quad (4.13)$$

where L_j is the length of the noisy signal of jth level. During threshold estimation, it is important to include the detail coefficients at each level as they have positive impact on the robustness of the threshold value. Therefore, the selected threshold has to be rescaled at a particular level. In this research work, the threshold value is estimated is dependent on the detail coefficients at every level.

- **Inverse Discrete Wavelet Transformation**

The inverse transform works on N data elements, where the first N/2 elements are smoothed values and the second N/2 elements are wavelet function values. The inner product that is calculated to reconstruct a signal value is calculated from two smoothed values and two wavelet values.

Logically, the data from the end is wrapped around from the end to the start. The pseudo code for D4 inverse transformation is given in Figure 4.7.

```
protected void invTransform( double a[], int n )
  {
    if (n >= 4)
   {
      int i, j;
      int half = n >> 1;
      int halfPls1 = half+1;
      double tmp[] = new double[n];

      //      last smooth val  last coef.  first smooth  first coef
      tmp[0] = a[half-1]*Ih0+a[n-1]*Ih1+a[0]*Ih2+a[half]*Ih3;
      tmp[1] = a[half-1]*Ig0+a[n-1]*Ig1+a[0]*Ig2+a[half]*Ig3;
      j = 2;
      for (i = 0; i < half-1; i++)
      {
        //     smooth val     coef. val       smooth val     coef. val
        tmp[j++] = a[i]*Ih0+a[i+half]*Ih1+a[i+1]*Ih2+a[i+halfPls1]*Ih3;
        tmp[j++] = a[i]*Ig0+a[i+half]*Ig1+a[i+1]*Ig2+a[i+halfPls1]*Ig3;
      }
      for (i = 0; i < n; i++)
           {   a[i] = tmp[i];   }
    }
  }
```

Figure 4.7 : D4 Inverse Transformation

4.1.3. Windowing

During analysis, it was found that the speech signal over short duration of time exhibit similar characteristics and the speech signal over long duration of time exhibit significant changes. Hence, short duration is preferred. For this reason, generally framing (or blocking) is used. In this method, the speech signal is divided into frames, which are termed as speech segments. A major issue with framing is that, in some cases, a feature of a single phoneme might be split into two frames, thus may not be effective. To overcome this problem,

in this research, a overlap windowing scheme (Gupta, 2006) is used and analysis is performed for each window. An example of overlap windowing is shown in Figure 4.8. The time for which the signal is considered is called a 'window' and the data in a particular window is called a 'frame'. This research work uses the most frequently used window in speech processing, namely, ‘Hamming Window’, as it introduces least amount of distortion.

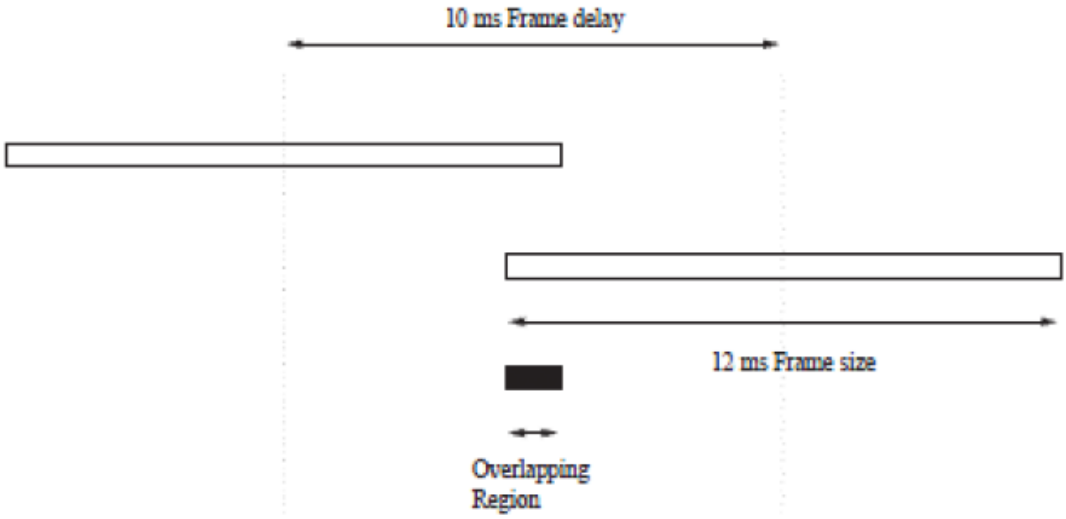


Figure 4.8 : Overlapping Windowing Scheme ***

4.1.4. Silence Removal

The second step of ATSQR system is silence removal and segmentation of speech signals from the input speech query signal. Any speech signal can have three types of speech states, namely, silence, noise and voice data (Sharma, 2012). Silence state has no sound produced and has no significant meaning during speech recognition and hence can be removed from further analysis. The noise data is handled using the W2SHT algorithm presented in the previous section. The third state has the actual speech data. To improve the performance of ATSQR system, only the speech data in third state need to considered.

The research work uses two criteria to identify the presence of spoken words in the query. They are the short time energy and zero crossing rate. Both of these were found to be necessary, as voiced sounds tend to have a high volume (and thus a high total energy), but a low overall frequency (and thus a low number of zero crossing), while unvoiced sounds were found to have a high frequency, but a low volume. Only background noise was found to have both low energy and low frequency. The algorithm is referred to as Silence Removal from Denoised Speech Tamil Query (SRDSTQ) and consists of the following steps.

1. Speech Analysis
2. Identification and removal of silence region

- **Speech Analysis**

The second step performs speech analysis to extract the two criteria, namely, short time energy and zero crossing rate, to identify speech data and silence data.

- **Short Time Energy (STE)**

The energy of a speech signal can be considered as a representation that reflects the amplitude variations over time. Consider a time speech signal x[n], the STE at sample n is defined as the summation of squared amplitude in a N-sample frame ending at n (Equation 4.14).

$$E_n = \sum_{m=n-N+1}^{n} (x[m])^2 \qquad (4.14)$$

The above Equation (4.14) is modified to Equation (4.15) to accommodate the windowing scheme used.

$$E_n = \sum_{m=-\infty}^{\infty} (x[m]\, w[n-m])^2 \qquad (4.15)$$

where n lies between zero and window size-1. The hamming window size was fixed as 50ms after empirical evaluation.

- **Zero Crossing Rate (ZCR)**

Zero crossing rate is defined as a measure of number of times the amplitude of speech signal passes through a value of zero in a given time interval or frame. It is a simple measure that reflects the frequency content of the speech signal. It has the advantages of being independent of speaker and speech volume. It can be estimated using Equation (4.16), where ZCR is estimated as the weighted average of the number of times the speech signal changes sign within the time window (Rabiner and Schafer, 2007).

$$Z_n = \sum_{m=-\infty}^{\infty} 0.5\,|\,sng\{x[m]\} - sgn\{x[m-1]\}w[\hat{n}-m] \qquad (4.16)$$

where x[m] is the speech sequence of the neighbourhood sample m at time $\hat{n}$ and w[$\hat{n}$ - m] is the time shifted window. The function sgn{x} is estimated using Equation (4.17).

$$sgn\{x\} = \begin{cases} 1 & x \geq 0 \\ -1 & x < 0 \end{cases} \qquad (4.17)$$

Since 0.5|sng{x[m]|-sng(x[m-1]}| is equal to 1 if x[m] and x[m-1] have different algebraic signs and 0 (zero) if they have the same sign, it follows that Z in is a weighted sum of all the instances of alternating sign (zero crossings) that fall within the support region of the time shifted window w[$\hat{n}$ - m]. During ZCR estimation, the hamming window size was set to 25ms after empirical experimentations.

- **Identification and removal of silence region**

The final step of the proposed SRDSTQ algorithm is the identification of the actual speech data. This step uses the above estimated STE and ZCR

along with a threshold value obtained automatically. This method was adopted from the submission of Giannakopoulos (2010). The automatic threshold estimation method is given below.

- Compute the histogram of the feature sequence‟s values.
- Detect the histogram's local maxima.
- Let M_1 and M_2 be the positions of the first and second local maxima respectively. The threshold value is computed using Equation (4.18).

$$T = \frac{W.M_1 + M_2}{W + 1} \tag{4.18}$$

where W is a user-defined weight parameter and was set to 5 as per the original author.

The above automatic estimation of threshold is executed for both feature sequences, to obtain two thresholds, namely, T1 and T2, for STE and ZCR respectively.

Using the above two thresholds, the silence and speech states are identified using a simple rule-based procedure. This procedure is given in Figure 4.9.

4.2. FEATURE EXTRACTION

After preprocessing, the second step of ATSQR is feature extraction. It is the process of converting the preprocessed speech waveform into a representation that allows analysis, comparisons and processing. The task of the feature extraction is to obtain the most relevant information from the original data and represent that information in a lower dimensionality space.

In the speech recognition community, two dominant feature extraction approaches are used. They are, temporal domain or parametric approach (Example Linear Prediction Cepstral Coefficient-LPCC) and non-parametric frequency domain approach (Example : Mel-Frequency Cepstral Coefficient - MFCC).

```
Input : Tamil Speech Query (SV)
Output Silence Removed Speech Query (SRSV)
Step 1  :  Read SV(n) where n is the length of the speech query
Step 2  :  Estimate STE and ZCR
Step 3  :  Compute thresholds, T1 and T2
Step 4  :  Let SRSV = {0}
Step 5  :  for i = 1 to n
        Step 5a : if STE > T1 and ZCR < T2 then // speech data
                SRVS[j]=SV[i]
                j=j+1
            elseif STE < T1 and ZCR < T2 then //silence data
                break
Step 6  :  Output SRVS
```

Figure 4.9 : Speech and Silence Identification

According to Rabiner and Juang (1993), LPCC is not suitable for representing speech data because it assumes signal stationary within a given frame and hence cannot analyze the localized events accurately. However, according to the results of Krishnan, *et al.* (1994), LPC analysis can distinguish words having distinct sounds. MFCC, on the other hand, has several advantages while used with speech data. These include low computational complexity and better performance with speech recognition. However, its performance is directly proportional to the presence of noise. According to Mallat (1998), wavelet-transform based features given better speech recognition accuracy than LPCC and MFCC. However, these techniques lose the time information due to the use of wavelet subband energies (Nehe and Holambe, 2012). Thus, all the three predominantly used feature extraction techniques have advantages as well as disadvantages.

In this research work, two feature extraction techniques that combine Discrete Wavelet Packet Transformation (DWPT) with LPCC and MFCC is proposed. This section first presents the basics of DWPT, conventional LPCC feature extraction, conventional MFCC feature extraction, followed by the enhanced versions of MFCC and LPCC feature extraction algorithms. During feature extraction, the hamming window size was fixed to 30ms frames.

4.2.1. Wavelet Packet Decomposition

Wavelet Packet Decomposition (WPD), also known as Optimal Subband Tree Structuring, wavelet packets or Subband Tree, is a wavelet transform where the discrete-time (sampled) signal is passed through more filters than the discrete wavelet transform (DWT) (https://en.wikipedia.org/wiki/Wavelet_ packet_decomposition).

In the DWT, each level is calculated by passing only the previous wavelet approximation coefficients (cA_j) through discrete-time low and high pass quadrature mirror filters (Coifman and Wickerhauser, 1992) . However, in the WPD, both the detail (cD_j (in the 1-D case), cH_j, cV_j, cD_j (in the 2-D case)) and approximation coefficients are decomposed to create the full binary tree (Benyassine and Akansu, 1995; Tazebay and Akansu, 1995). Figure 4.10 shows a 3-level WPD, where g[n] is the low-pass approximation coefficients and h[n] is the high-pass detail coefficients

For ‘n’ levels of decomposition, the WPD produces 2^n different sets of coefficients (or nodes) as opposed to (3n + 1) sets for the DWT. However, due to the downsampling process the overall number of coefficients is still the same and there is no redundancy.

4.2.2. Conventional MFCC Feature Extraction

A block diagram of the structure of the conventional MFCC feature extraction technique (Bharathi *et al.*, 2006) is given in Figure 4.11. The speech input is typically recorded at a sampling rate above 10000Hz. The sampling

frequency was chosen to minimize the effects of aliasing in the analog-to-digital conversion.

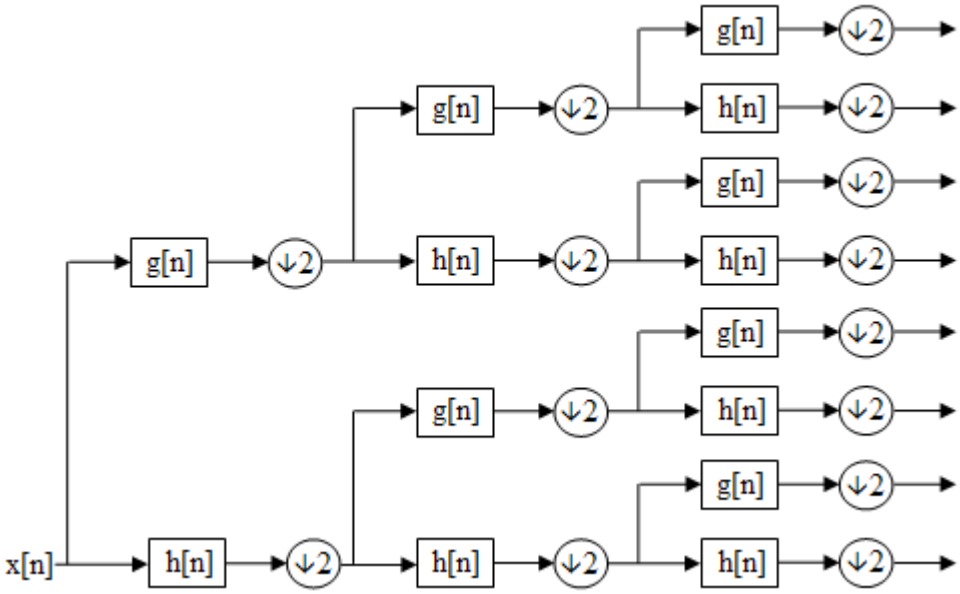


Figure 4.10 : 3-level Wavelet Packet Decomposition

The sampled signals capture all frequencies upto 5 kHz, which cover most energy of sounds that are generated by humans. The main purpose of the MFCC processor is to mimic the behavior of the human ears and MFCC's are less susceptible to variations. The following steps are involved in MFCC processor for generating MFCC.

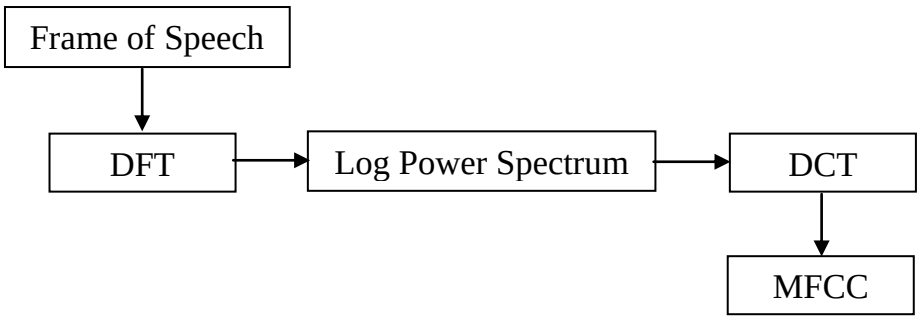


Figure 4.11 : MFCC Feature Extraction Process

During frame blocking, the speech signal is blocked into frames of N samples, with adjacent frames being separated by M (M < N). The first frame consists of the first N samples. The second frame begins M samples after the

first frame, and overlaps it by N – M samples. This process continues until all the speech is accounted for within one or more frames. Typical values for N=256 (which is equivalent to ~ 30 msec windowing and facilitate the fast radix-2 FFT) and M=100.

The next step in the processing is to window each individual frame so as to minimize the signal discontinuities at the beginning and end of each frame. In this step, the hamming window described in Section 4.1.3 is used.

The next step is the Fast Fourier Transform, which converts each frame of N samples from the time domain into frequency domain. The FFT is a fast algorithm to implement the Discrete Fourier Transform (DFT), which is defined on the set on N samples $\{x_n\}$ as follows:

$$X_n = \sum_{k=0}^{N-1} k^{e^{-2\Pi_j k_n / N}} \qquad (4.19)$$

where j denotes the imaginary unit, i.e. $j = \sqrt{-1}$, x_n's are complex numbers. The resulting sequence $\{x_n\}$ is interpreted as follows:

- The zero frequency corresponds to n= 0, positive frequencies $0 < f < Fs/2$ corresponds to values $1 \leq n \leq N/2-1$
- Negative frequencies $-Fs/2 < f < 0$ correspond to $N / 2 + 1 \leq n \leq N-1$

Here, Fs denote the sampling frequency. The result after this step is often referred to as spectrum or periodogram.

The human perception of the frequency contents of sounds for speech signals does not follow a linear scale. Thus for each tone with an actual frequency f, measured in Hz, a subjective pitch is measured on a scale called the 'mel' scale. The mel-frequency scale is linear frequency spacing below 1000Hz and a logarithmic spacing above 1000Hz. As a reference point, the pitch of 1 kHz tone, 40 dB above the perceptual hearing threshold, is defined as

1000 mels. The following approximate formula (Equation 4.20) is used to compute the mels for a given frequency f in Hz:

$$\text{Mel}(f) = 2595 * \log_{10}(1 + f / 700) \quad (4.20)$$

In this final step, the log mel spectrum is converted back to time. The result is called the mel frequency cepstrum coefficients (MFCC). The cepstral representation of the speech spectrum provides a good representation of the local spectral properties of the signal for the given frame analysis. As the mel spectrum coefficients are real numbers, it is converted to time domain using the Discrete Cosine Transform (DCT). $\tilde{S}_k$, k=1.2…K, denotes those mel power spectrum coefficients then the MFCC's, $\tilde{C}_n$, are calculated as follows:

$$\tilde{C}_n = \sum_{k=1}^{K} (\log \tilde{S}_k) \cos[n(k-1/2)\prod/K] \quad (4.21)$$

where n = 1,2, …, K. The first component, $\tilde{C}_0$, is excluded from the DCT since it represents the mean value of the input signal, which carries little speaker specific information.

4.2.3. Proposed Wavelet-Packet-Based MFCC Feature Extraction Technique

The proposed enhanced version of MFCC feature extraction technique replaces DFT block in Figure 4.12 with DWPT to obtain the advantages of wavelet packet transformation. This technique is referred to as WPMFCC in this research. The coefficients from the wavelet packets are subjected to log power spectrum and then Discrete Cosine Transformation (DCT). For Each WPMFCC computed from the DWT coefficients, only the first 10 coefficients are concatenated to obtain the final feature vector. The steps are presented in Figure 4.12 and the process of obtaining WPMFCC is presented in Figure 4.13.

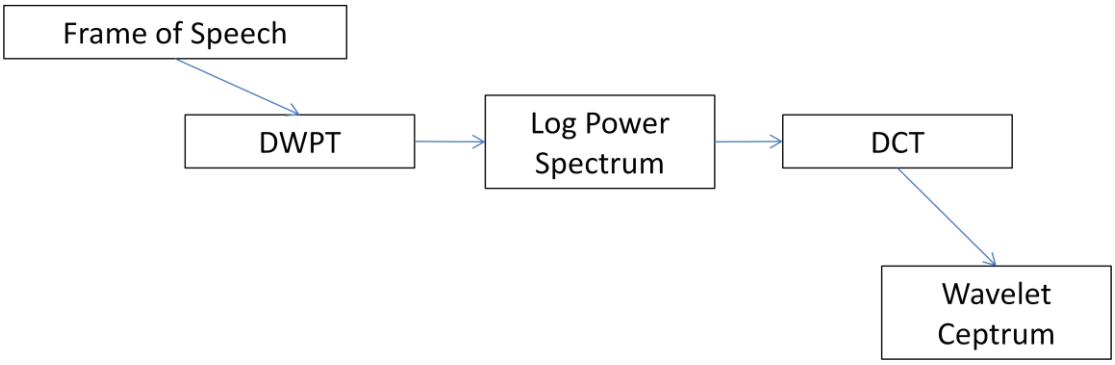


Figure 4.12 : DPMFCC Feature Extraction Process

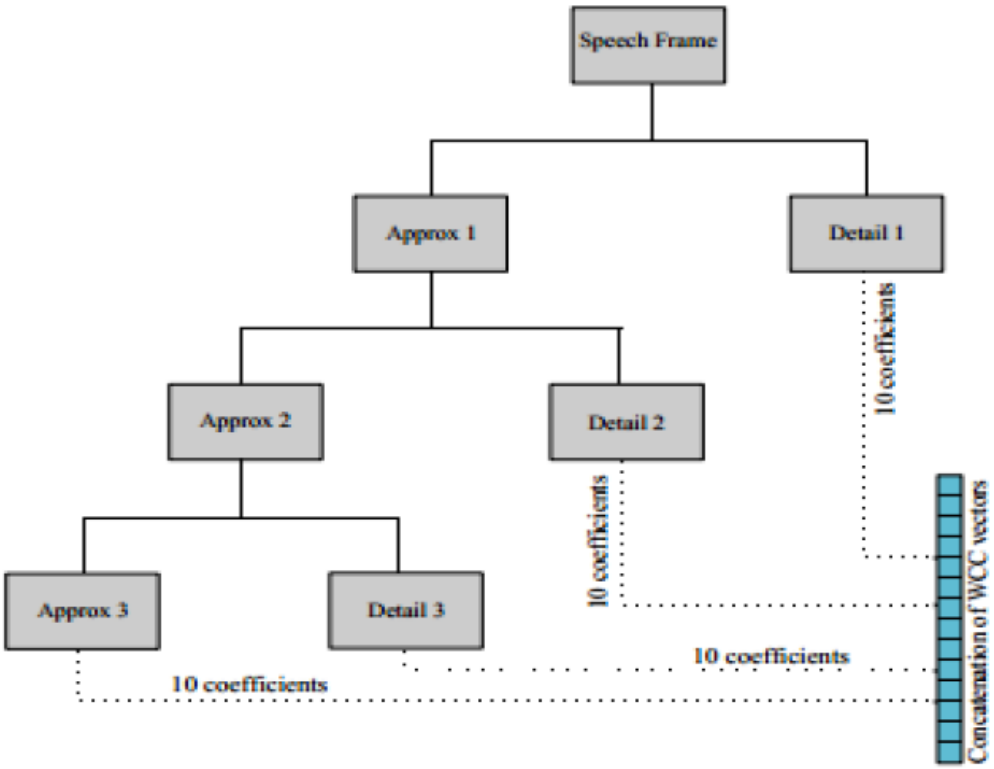


Figure 4.13 : Process of Obtaining WPMFCC Features from WPD

It was noticed that recognizer performance degraded because of variability in the acoustic realization of the utterance, which could come from various sources. First, it may be due to change in the environment as well as position and characteristics of the microphone. Second with in speaker, variability could result from change in speaker's physiological state, speaking rate, voice quality, socio-linguistic background, dialect, vocal tract size, shape etc. These factors contribute to change in amplitude, duration and SNR ratio for a given utterance. Hence in order to make the system more robust to above said

distortions a normalization technique is implemented by which cepstral coefficients will be normalized to have zero mean and unit variance within the given frame.

The normalization coefficients were calculated over a relatively short sliding window (frame). The feature vectors were normalized as follows:

$$C'_{t-T(j)} = [C_{t-D}(j) - \mu_t(j)] / \sigma_t(j) \quad (4.22)$$

where $C_{t-D}(j)$ is the jth component of the original feature vector at time t – T, $C'_{t-T}(j)$ is the normalized version and T denotes the delay in terms of feature vectors. The normalization coefficients, mean $\mu_t(j)$ (Equation 4.23) and standard deviation $\sigma_t(j)$ (Equation 4.24). For each feature vector component j were calculated over the sliding finite length normalization window as shown below.

$$\mu_t(j) = \frac{1}{N \sum_{n=1}^{N} C_n(j)} \quad (4.23)$$

$$\sigma_t(j) = \frac{1}{N \sum_{n=1}^{N} (C_n(j) - \mu_t(j))^2} \quad (4.24)$$

where N denotes the normalization segment length in terms of the feature vectors. Here the mean removal can be regarded as the linear high pass filter and division by standard deviation act as an automatic gain control.

4.2.4. Conventional LPCC Feature Extraction

LPC analysis estimates vocal tract resonance called pitch formants and spectra; in order to get source signal it removes the effect of speech signal called as residual signal. It finds unknown sample parameters from linear combination of past samples of speech. Before feature extraction speech waveform must undergo preprocessing technique like pre-emphasis, frame

blocking and Windowing operations. The block diagram of linear predictive cepstral coefficients feature extraction technique is shown in Figure 4.14.

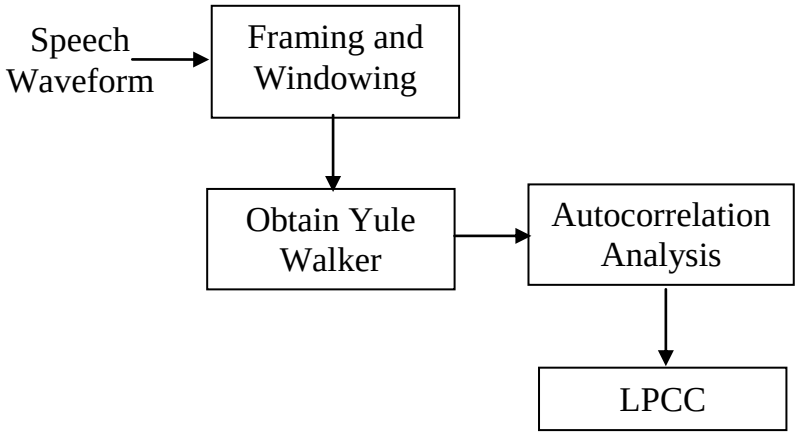


Figure 4.14 : Block diagram of LPCC feature Extraction

Pre-Emphasis and windowing were performed using procedures mentioned in Sections 4.1.1 and 4.1.3. The actual and linear predictions of samples are taken to minimize their sum of squared differences within the finite interval to obtain predictor coefficients. These predictive coefficients are transformed into robust set of cepstral coefficients which is useful for linear predictive analysis of speech. The difference equations describing the relation between speech samples s[n] and excitation signal u(n) is as follows.

$$s[n] = \sum_{i=1}^{p} a_i \, s(n-i) + G \, u[n] \qquad (4.25)$$

The system function in form

$$H(z) = \frac{G}{1 - \sum_{i=1}^{P} a_i z^{-i}} \qquad (4.26)$$

A linear predictor of order p with prediction coefficients a_i is defined using Equation (4.27).

$$\overline{s[n]} = \sum_{i=1}^{p} a_i \, s(n-i) \qquad (4.27)$$

The prediction error $e_m(n)$ is defined as

$$e_m(n) = s[n] - \overline{s[n]} \quad (4.28)$$

Comparing equation (4) and (6) if ai=ai then inverse filter A(z) prediction error filter.

$$A(z) = 1 - \sum_{i=1}^{p} a_i z^{-i} \quad (4.29)$$

To obtain predictor coefficients, minimize the mean squared difference of prediction error is given by

$$E_m = \sum_n (e_m[n])^2 = \sum_n (x_m[n] - \sum_{j=1}^{p} a_j x_m[n-j])^2 \quad (4.30)$$

Here, $X_m[n]$ is speech frame of signal and p is the order of LPC analysis. Estimating LPC coefficients by assuming orthogonality principle Yule walkers equation given as

$$\sum_{j=1}^{p} a_j \varphi_m[i,j] = \varphi_m[i,0] \ , \qquad i = 1,2 \ldots . p \quad (4.31)$$

The values of a_j that minimize E_m , by setting $\delta E_m / \delta a_j = 0$, j=1,2…..p.

$$\varphi_m[i,j] = \sum_n x_m[n-i] x_m[n-j] \quad (4.32)$$

The above problem is solved by different methods like autocorrelation, covariance and lattice method. Here autocorrelation method is used.

$$r = R \times a \quad (4.33)$$

where R is autocorrelation matrix, ‘a’ is LPC coefficient and ‘r’ is autocorrelation vector. The resulting equation is solved by Levinson Durbin’s recursive procedure. The procedure works recursively and finally, all the prediction coefficients were obtained of order less than p.

4.2.5. Proposed Wavelet-Packet-Based LPCC Feature Extraction Technique

The proposed WPD-based LPCC feature extraction technique is termed as WPLPCC in this research. Figure 4.15 shows the steps involved in WPLPCC feature extraction technique.

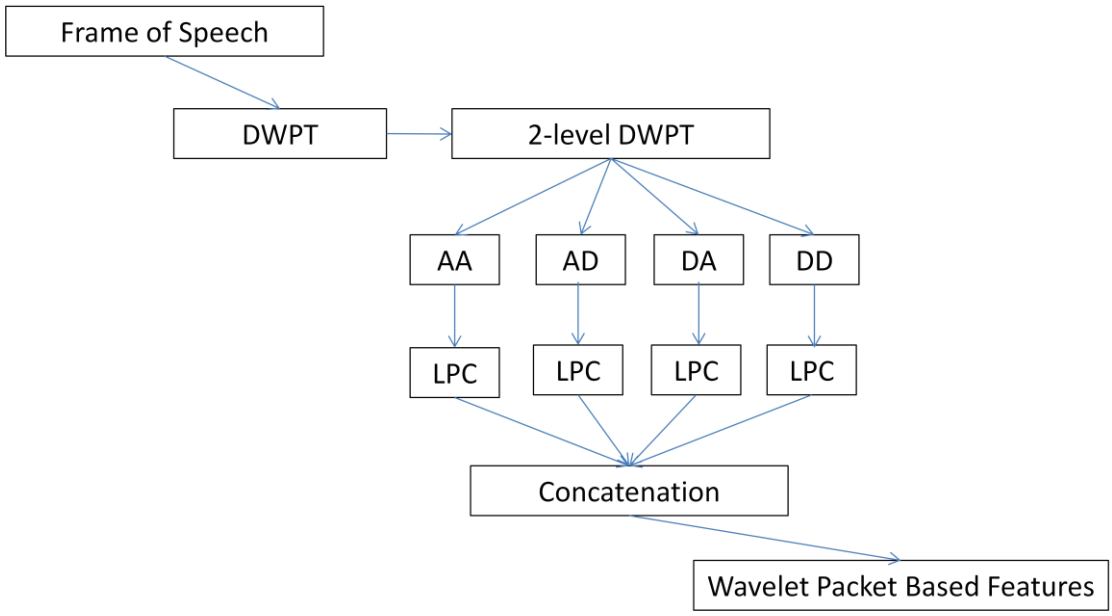


Figure 4.15 : Steps in WPLPCC Feature Extraction Technique

The algorithm first performs a 2-level WPD to obtain four subbands, namely, AA, AD, DA and DD. The LPC technique is applied on each subband signal to obtain the combined benefits of Wavelet packets and LPC. The LPC features of pth order have been extracted frome ach subband of wavelet decomposed speech signal. The obtained LPC coefficients are then concatenated to form the final feature vector. The feature vector f_i from frame i is expressed using Equation (4.34).

$$f_i = [AA\ AD\ DA\ DD]^T \tag{4.34}$$

where AA, AD, DA, DD are the approximate and detailed coefficients obtained after level 2 wavelet packet decomposition and T indicates a vector transpose.

After obtaining the final feature vector, cepstral mean and variance normalization using CMN (Rosenberg *et al.*, 1994) feature normalization algorithm was performed to obtain coefficients whose mean of the cepstral sequence is zero, and it has a variance of one.

4.3. SPEECH QUERY RECOGNITION

The proposed ATSQR system uses an ensemble SVM classifier to recognize the Tamil speech query words. As this classifier cannot work with variable sized feature vector, an algorithm using Self Organizing Map is proposed to convert the variable sized feature vector to a constant size. The SOM algorithm uses as input the variable length feature vector and maps it to a constant size of six clusters while preserving the input size. The algorithm consists of three tasks, namely, Competitive task, Cooperative task and Adaptation task. Detailed description of this algorithm can be obtained from Sampath *et al.* (2010). This section presents details of the conventional SVM and ensemble classifiers.

4.3.1. Support Vector Machine (SVM)

Support Vector Machines (SVMs) were introduced by Vapnik (1995) (Basu *et al.*, 2002) and have proved to be fast effective classifiers and works effectively with high dimensional datasets also (Eirinaki, 2009). A Support Vector Machine (SVM) is a concept in computer science for a set of related supervised learning methods that analyze data and recognize patterns that are mainly used for classification and regression analysis. The standard SVM takes a set of input data and predicts, for each given input, which of two possible classes the input is a member of, which makes the SVM a non-probabilistic binary linear classifier.

Given a set of training examples, each marked as belonging to one of two categories, an SVM training algorithm builds a model that assigns new examples into one category or the other. An SVM model is a representation of

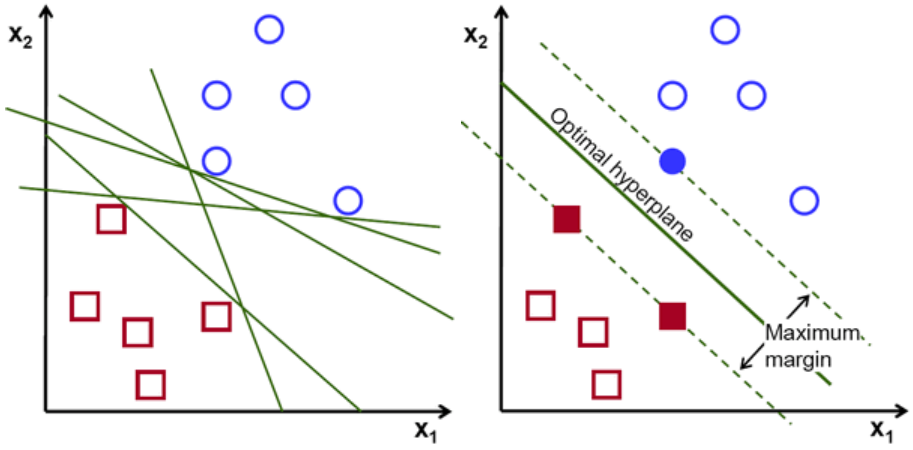
the examples as points in space, mapped so that the examples of the separate categories are divided by a clear gap that is as wide as possible. New examples are then mapped into that same space and predicted to belong to a category based on which side of the gap they fall on.

Although SVMs were originally designed as binary classifiers, approaches that address a multi-class problem as a single “all-together” optimization problem exist (Weston and Watkins, 1999). A multi-class classification task usually involves separating data into training and testing sets. Each instance in the training set contains one ‘target value" (i.e. class labels) and several “attributes" (i.e. features). The goal of SVM is to produce a model (based on the training data) which predicts the target values of the test data given only the test data attributes. Mathematically SVM can be described as follows (Cortes and Vapnik, 1995).

Considering the binary classification case, let $((x_1, y_1) \ldots (x_n, y_n))$ be the training dataset where xi are the feature vectors that represent the observations and $y_i \in (-1, +1)$ be the two labels that each observation can be assigned to. From these observations, SVM builds an optimum hyperplane (a linear discriminant in the kernel transformed higher dimensional feature space) that maximally separates the two classes by the widest margin by minimizing the objective function. For a linearly separable set of 2D-points which belong to one of two classes, find a separating straight line is shown in 4.16a. In this example, there exist multiple straight lines that separate the data points into two groups. Deciding the optimal divider is an intuitive criterion.

In general, a line is considered bad if it passes too close to the points because it will be noise sensitive and it will not generalize correctly. Therefore, the goal here is to find the line passing as far as possible from all points. Thus, the operation of the SVM algorithm is based on finding the hyperplane that gives the largest minimum distance to the training examples. Twice, this distance receives the important name of margin within SVM’s theory.

Therefore, the optimal separating hyperplane maximizes the margin of the training data. Example of an optimal hyperplane is shown in Figure 4.16b.



(a) **(b)**

Figure 4.16 : Support Vector Machine Hyperplane

- **Hyperplane Computation**

Let the hyperplane be defined as

$$f(x) = \beta_0 + \beta^T x \qquad (4.35)$$

where β is known as the weight factor and β_0 is the bias. The optimal hyperplane is represented in an infinite number of different ways by scaling of β and β_0.

As a matter of convention, among all the possible representations of the hyperplane, the one chosen is

$$| \beta_0 + \beta^T x| = 1 \qquad (4.36)$$

where x symbolizes the training examples closest to the hyperplane. In general, the training examples that are closest to the hyperplane are called support vectors and this representation is known as the canonical hyperplane. Now, the result of geometry that gives the distance between a point x and a hyperplane $\{\beta, \beta_0\}$ is estimated using Equation (4.37).

$$\text{distance} = \frac{|\beta_0 + \beta^T x|}{||\beta||} \tag{4.37}$$

In particular, for the canonical hyperplane, the numerator is equal to one and the distance to the support vectors is Equation (4.38).

$$\text{distance}_{\text{support\_vectors}} = \frac{|\beta_0 + \beta^T x|}{||\beta||} = \frac{1}{||\beta||} \tag{4.38}$$

Let M denote the margin which is twice the distance to the closest examples (Equation 4.39).

$$M = \frac{2}{||\beta||} \tag{4.39}$$

Now, the problem of maximizing M is equivalent to the problem of minimizing a function L(β) subject to some constraints. The constraints model the requirement for the hyperplane to classify correctly all the training examples x_i Formally, it can be defined as Equation (4.40).

$$\min_{\beta,\beta_0} L(\beta) = \frac{1}{2}||\beta||^2 \qquad \text{subject to } y_i(\beta^T x_i + \beta_0) \geq 1 \ \forall i \tag{4.40}$$

where y_i represents each of the labels of the training examples.

SVM parameters are set as given below.

SVM_Params={'-b 0 -c 100 -d 3 -g 1 -t 3 '}

Here, -b indicates the probability estimates and takes the values 0 or 1 to indicate whether to train a SVC (Support Vector Classification) or SVR (Support Vector Regression) model for probability estimates, 0 or 1.The study uses SVR. To allow some flexibility in separating the categories, SVM models have a cost parameter, C, that controls the tradeoff between allowing training errors and forcing rigid margins. It creates a soft margin that permits some

misclassifications. Increasing the value of C increases the cost of misclassifying points and forces the creation of a more accurate model that may not generalize well. The cost parameter is set to 100. The next parameter, d, is degree in kernel function which is assigned a value 3, while g (gamma in kernel function) is assigned a value of 1.

The last parameter '-t' is used identify the kernel function to be used. The kernel function, $K(x_i, xj) \equiv (x_i)^T \phi(x_j)$, can belong to any one of the following four types available.

1. Linear Kernel : $K(x, x_j) = x_i^T x_j$
2. Polynomial Kernel : $K(x_i, x_j) = (\gamma x_i^T x_j + r)^d, \gamma > 0.$
3. Radial Basis Function (RBF) : $K(x_i, x_j) = \exp(-\gamma||x_i - x_j||^2), \gamma > 0.$
4. Sigmoid Kernel : $K(x_i , x_j) = \tanh(x_i^T x_j + r).$

Here, γ, r and d are kernel parameters. The study uses the radial basis function.

4.3.2. Ensemble SVM-Based Classifier

According to Neeba and Jawahar (2009), the performance of a classification system can be improved by using multiple single classifiers and then use algorithms to combine the output of single classifiers. Multiple classifiers are defined as the usage of different types of classifiers or different instantiations of the same classifier. This type of classification system is termed as 'Ensemble Classification' or 'Fusion Classification'. This research work uses ensemble classification to recognize the words of the Tamil speech query.

According to Park *et al.* (2010), when a perfect set of features that can describe a spoken word is given, the accuracy of the resultant classification depends on the classifier adopted. Several solutions have been proposed for this purpose. Predominant among these is the Support Vector Machine (SVM). The reasons behind this popularity are its easy implementation procedures and accurate classification. To construct an ensemble SVM model, a homogeneous

method is used. Homogeneous classifiers are considered as those models having the same classification methodology but different feature vectors, while heterogeneous classifiers are models with each classifier using a different classification methodology. In this research work, the Adaboost subspace selection algorithm is used to create the SVM-based ensemble system. This system consists of two steps, namely, (i) Training the component classifiers and (ii) Combining the results. The performance of the SVM ensemble classifier depends on the following factors.

(i) Feature vector
(ii) Ensemble creation method Partitioning method
(iii) Aggregation method
(iv) Type of training

- **Feature Vector**

The research work analyzes the applicability and benefits obtained by four feature vectors namely,

(i) MFCC-based Features
(ii) LPCC-based Features
(iii) WPMFCC-based Features
(iv) WPLPCC-based Features

- **Ensemble Creation Method**

During the creation of the proposed SVM ensemble classifier, as mentioned earlier, the Adaboost subspace selection method is used. Boosting (Freund and Shapire, 1995, 1996) induces the ensemble of learners by adaptively changing the distribution of the training set based on the performance of the previously created classifiers. Boosting relies on the concept of under-sampling the original training set. The sampling process is adaptive and depends on the performance of previously created predictors. The

weight of each pattern changes dynamically as the algorithm runs, according to the performance of past predictors. Boosting tends to give larger weights on patterns that are misclassified by previous classifiers. These hard patterns have a higher chance of being picked up by future predictors. Therefore, later classifiers focus on these hard patterns in an attempt to learn them more accurately. Figure 4.17 presents the boosting algorithm, called AdaBoost (ADAptive BOOSTing), and used to create the ensemble system.

AdaBoost adaptively adjusts the distribution of the original training patterns. It starts by setting all the patterns with equal probability of being chosen in the bootstrap sample. Then, for each predictor, it generates a different training set by sampling, with replacement, from the original set, according to the weight distribution of the patterns. For each created predictor, the output of each pattern is computed, and a relative loss function is computed. The weight of each pattern is then adapted according to its difficulty i.e. the more incorrect the predictor was, the larger the weight it takes. The weights are then renormalized to remain a valid distribution. Success of Boosting algorithm has been proved by several studies (Quinlan *et al.,* 1996; Zheng and Webb, 1998). It can generate, on average, very accurate ensembles using many learning algorithms for classification.

- **Partitioning Method**

The next step in the design of ensemble classifier is to decide the partitioning method that will be used to create the training and testing feature set. In this research work, the hold-out method is used for this purpose. The holdout method randomly partitions the dataset into two independent sets, training (generally two-thirds of the data) and testing (generally one-third of the data). The method is pessimistic because only a portion of the initial data is used to derive the model.

Inputs: Set S of m training feature: S ={(x_i,y_i), i = 1, . . . ,m}, where x_i is the

Ith vector of features and y_i is the ith target label, LEARN: a learning algorithm , K : the number of classifiers to return (=25)

Outputs:

An ensemble E of K predictors: E = {C_k, k = 1, . . . ,K}

Initialize: $w_i^{(1)} = \frac{1}{\sum_{i=1}^{m} w_i}$ {initialize weights of training set}

Procedure:

for k := 1, . . . ,K do {loop on number of classifiers}

construct a training set T_k using the distribution w(k)

C_k :=LEARN(T_k) {train classifier C_k using bootstrap sample T_k}

$\hat{y}_i^{(k)} = C_k(x_i)$ for all i, i = 1, . . . , m

$L_i = \frac{\left|\hat{y}_i^{(k)} - y_i\right|}{D}$ for all i and D = $\max_{i=1,\ldots,m} |\hat{y}_i^{(k)}$ {loss for each feature}

$\bar{L} = \sum_{i=1}^{m} L_i w_i^{(k)}$ {average loss function}

$\beta_k = \frac{L}{1-L}$ {confidence measure}

$w_i^{(k+1)} = w_i^{(k)} \beta_k^{(1-L_i)}$

$w_i^{(k+1)} = \frac{w_i^{(k+1)}}{\sum_i w_i^{(k+1)}}$ {normalize weight so that it remains a distribution}

end for

Figure 4.17 : Boosting Algorithm

- **Aggregation Method**

While using multiple classifiers, a method that combines the results of the various classifiers is needed. Several techniques exists, namely, majority voting, maximum, sum, min, average, product, Bayes, decision template and behavior knowledge space. This research work uses majority vote scheme, which is one of the oldest strategies for decision making. It is the most frequently used method because of its simplicity and speed. The method is explained below.

Let the decision of the i^{th} classifier be defined as $d_{t,j} \in \{0, 1\}$, t = 1, …, T and j = 1, …, C, where T is the number of classifiers and C is the number of classes. If the i^{th} classifier chooses class ω_j, then $d_{t,j}$ = 1 and 0, otherwise. In majority voting scheme, a class ω_j is chosen, if

$$\sum_{t=1}^{T} d_{t,J} = \max_{j=1}^{c} \sum_{t=1}^{T} d_{t,j} \qquad (4.41)$$

With majority voting, the fusion classifier makes the correct decision if at least $\lfloor T/2 \rfloor + 1$ classifiers choose the correct label, where the floor function $\lfloor . \rfloor$ returns the largest integer less than or equal to its argument. The accuracy of the fusion classifier can be represented by the binomial distribution as the total probability of choosing $k \geq \lfloor T/2 \rfloor + 1$ successful ones out T of classifiers, where each classifier has the success rate of p. Hence, P_{ens}, the probability of fusion classification success is

$$P_{ens} = \sum_{k=(T/1)+1}^{T} \binom{T}{k} p^k (1-p)^{T-k} \qquad (4.42)$$

In majority voting, each classifier has the same weight and voting by ensemble members determines the final result. Usually, it takes over half of ensemble members to agree in order for a result to be accepted as the final output of the

ensemble, regardless of diversity and accuracy of each classifier's generalizability

- **Type of Training**

There are various methods used while training a multiple classifier system.

- Training of the individual classifiers and applying aggregation that does not require further training (e.g., aggregation techniques like average, minimum, product, maximum, etc.)
- Training of the individual classifiers followed by training the aggregation
- Simultaneous training of the whole scheme.

The present scheme uses the first method. This method is selected because the ensemble classification depends on the result of the individual classifier.

4.4. CHAPTER SUMMARY

This chapter presented the speech recognition system used to convert the Tamil speech query to text. For this purpose, the proposed algorithm performed preprocessing to remove noise and silence speech data, followed by MFCC and LPCC feature selection based on wavelet packets and finally used an ensemble SVM classifier created using Adaboost algorithm. The second phase of the research work focuses on the development of the query translation system, which converts the Tamil Query to English Query. The methods used during the various steps of query translation are presented in the next chapter, Chapter 5, Design of Query Translation System.