The 1996 Mississippi State University Conference on

Speech Recognition

What:	EE 8993 Project Presentations
Where:	432 Simrall, Mississippi State University
When:	May 1, 1996 — 1:00 to 4:00 PM

SUMMARY

The Department of Electrical and Computer Engineering invites you to attend a mini-conference on Speech Recognition, being given by students in EE 8993 — Fundamentals of Speech Recognition. Papers will be presented on a wide range of topics including signal modeling, hidden Markov models (HMMs), search algorithms, and language modeling.

Students will present their semester-long projects at this conference. Seven students will give 10 minute presentations, followed by 5 minutes of discussion. After the talks, each student will be available for a live-input real-time demonstration of their project. These projects account for 100% of their course grade, so critical evaluations of the projects are encouraged.

Session Overview

1:00 PM — 1:10 PM:	J. Picone, Introduction
1:15 PM — 1:30 PM:	**J. Trimble,** “Front-end of a Speech Recognizer”
1:30 PM — 1:45 PM:	**L. Webster**, “Front End Modeling with Special Emphasis on FFTs, LPC, and Feature Selection”
1:45 PM — 2:00 PM:	**R. Seelam**, “Implementation of Statistical Modeling Techniques and Channel Adaptation Techniques”
2:00 PM — 2:15 PM:	**A. Ganapathiraju**, “Implementation of Viterbi Beam Search Algorithm”
2:15 PM — 2:30 PM:	**N. Deshmukh**, “Efficient Search Algorithms for Large Vocabulary Continuous Speech Recognition”
2:30 PM — 2:45 PM:	**O. LaGarde**, “Language Modeling and Grammar Construction for an HMM Continuous Speech Recognition System”
2:45 PM — 3:00 PM:	**S. Given**, “Development of an N-Gram Based Language Model for Continuous Speech”
3:00 PM — 4:00 PM:	Demonstrations in 434 Simrall

AUTHOR INDEX

Volume I

Speech Recognition

Table of Contents

proposal for

Front-end of a Speech Recognizer

submitted to fulfill the semester project requirement for

EE 8993: Fundamentals of Speech Recognition

February 1, 1996

submitted to:

Dr. Joseph Picone

Department of Electrical and Computer Engineering
413 Simrall, Hardy Rd.
Mississippi State University
Box 9571
MS State, MS 39762

submitted by:

J. Trimble III

Department of Electrical and Computer Engineering
Mississippi State University
Box 9571
Mississippi State, Mississippi 39762
Tel: 601-325-3149
Fax: 601-325-2298
email: trimble@isip.msstate.edu

I. ABSTRACT

This proposal describes a plan to design and implement the front-end of a speech recognition sytem. The front end must derive a smooth spectral estimate of a signal in order to produce feature vectors that are compatible with the acoustic models of the system. Linear prediction provides an efficient and simple means of computing these feature vectors. Its basic purpose is to as accurately predict currents values of a signal based on a weighted sum of the signal's previous values. In addition, an even better spectral estimator, cepstral analysis, will be implemented.

II. INTRODUCTION

Speech is a very natural and an efficient manner of communication for humans. The study of speech recognition concerns itself with the development of systems that emulate the human's natural ability to process such an exchange of information. However, researchers have found that this is not a simple task.

In the early 1970s, commercial systems were developed to recognize isolated words. These were small systems which employed a vocabulary of 10 to 100 words. Towards the late 1980s, researchers at IBM developed a system capable of recognizing continuous (naturally spoken) and isolated utterances from a vocabulary of 20,000 and 5000 words respectively. Today, continuous speech recognition systems are not uncommon. Present systems can achieve recognition with an error rate of 5% to 10%.

All systems perform better when required only to recognize a single speaker who is used to "train" the system. However, whether it be a single or multi-speaker system, it's level of success is dependent upon the speaker's clear articulation and the systems's robustness to noise. Therefore, for a system to be maximally useful, it must achieve recognition of continuous speech, recognition of multiple speakers, and recognition within noisy environments. It should also be cost effective and should run in real time.

The speech recognition system can be broken down into three main segments--the front end, the language model, and the decoder (search algorithm). The front end converts the speech signal into a sequence of feature or acoustic vectors. Typically, a 10ms frame is used to segment the signal into blocks that will then be used to produce a spectral representation of the signal. The front end may employ different techniques to generate the acoustic vectors. These include Linear Prediction, Fast Fourier Transforms, and various transformations. The language model determines the probability of a word given a set number of proceeding words, and the decoder performs extensive searches in order to find the best possible word sequence as the recognizer's output.

Through the use of acoustical analysis, the vocal tract can be modelled by a concatenation of lossless tubes, each having a specific resonant frequency. Spectrum plots of vowel and consonant sounds reveal three main formants that normally appear. Due to these spectrum shapes, a concatenation of only three lossless tubes suffices to represent the three formants. As a result, researchers use the first three formants of the phone spectrums to model speech. With the use of the above tube models, a simple all pole system function for speech production can be derived.

Consequently, these poles model the acoustic resonances found within phones, especially vowels. Phenomenons such as lip radiation, nasals, and fricatives introduce zero into our system function. So why not use a zero-pole model for speech production? As later discussion will show, the all pole model lends itself to computing the system function coefficients with simple linear equations.

III. PROJECT SUMMARY

The scope of this project will only involve the front-end of the overall speech recognizer. Eventually, the final product will be integrated within an entire speech recognition system whose different components will be developed in parallel. Though the front-end may be implemented with various techniques, linear prediction based measurements will be the primary spectral estimator used. Cepstral analysis will also be included.

The all pole model of speech production mentioned earlier is a minimum phase representation of the speech signal. The absence of phase does not hinder the recognizer at all. In fact, human ears are deaf to phase. All information within speech is contained solely in the signal's magnitude. Therefore, a linear prediction representation of the speech model is not only simple but works well.

The input speech signal will be sampled at a minimum 8 KHz with a digital audio tape. A sample frequency of 8 KHz is chosen because the first three formants of voiced phone spectrums fall below about 3.6 KHz. The program should allow the user to adjust the sample frequency and the frame and window durations. The signal will then be fed into a preemphasis filter to magnify the higher frequencies. In doing so, one or two formants may be thrown away since the lower resonances usually fall within a general range. Next, windowing the signal will allow us to regard the speech signal as stationary within the interval of the window. The following step involves the linear prediction process.

Given a speech signal, one would like to predict the current value of the signal based on a linearly weighted sum of its past values. In statistical terminology, the output is said to regress on itself. The basic purpose of linear prediction is to minimize the error between the speech signal and the predicted signal with respect to the weights or predictor coefficients. Through mathematical analysis, these predictor coefficients may be computed by two methods (the covariance method and the autocorrelation method). The difference between the two is that for the autocorrelation method only data within the frames are used. The autocorrelation matrix is even and thus always has an inverse. Given that the linear prediction process can be modelled as a filter and the fact that the same coefficients satisfy both methods, the result is a minimum phase filter. An efficient algorithm that will be used for calculating the filter or predictor coefficients is the Levinson-Durbin recursion. The algorithm generates intermediate values called reflector coefficients that are used to calculate the predictor coefficients. The resulting filter is called an analyzer. Feeding the input speech signal into the this filter produces an error signal. More over, feeding the error signal into the inverse of the filter (synthesizer) produces the original input signal.

One drawback to the linear prediction process is that its measurements include the glottal excitation within the vocal tract impulse response. Because glottal excitation is different for each person, damaging information may be introduced into the system, especially for speaker-independent systems. A solution to such a problem is obtained through cepstral analysis which will also be implemented.

Cepstral analysis is a very useful deconvolution tool that addresses the problem of the nonlinear combination of the glottal excitation and the vocal tract impulse response. The cepstrum (analogous to a spectrum) represents the transformation on the speech signal. The cepstrum's properties allow it to represent the components of the signal separately as a linear combination. This is done by taking a base 10 log of the signal's short-term DFT and then applying an IDFT to the entire result. Remembering that convolution in the time domain gives rise to multiplication in the frequency domain, taking a log of a product produces a sum of logs. Therefore, the glottal excitation of the signal may be separated from the vocal tract response. Actually, it can be eliminated entirely through low-pass filtering.

The short-term real cepstrum was previously described. Through a transformation, the cepstral coefficients may also be derived from the linear prediction coefficients.

IV. TESTING

Testing will be carried out in conjunction with the different implementations of the front-end. 8 KHz, 16 bit data will be used as input. The implementation of the linear prediction process will be evaluated by inserting the error signal into the synthesizer in order to retrieve the original speech signal.

V. PLAN OF WORK

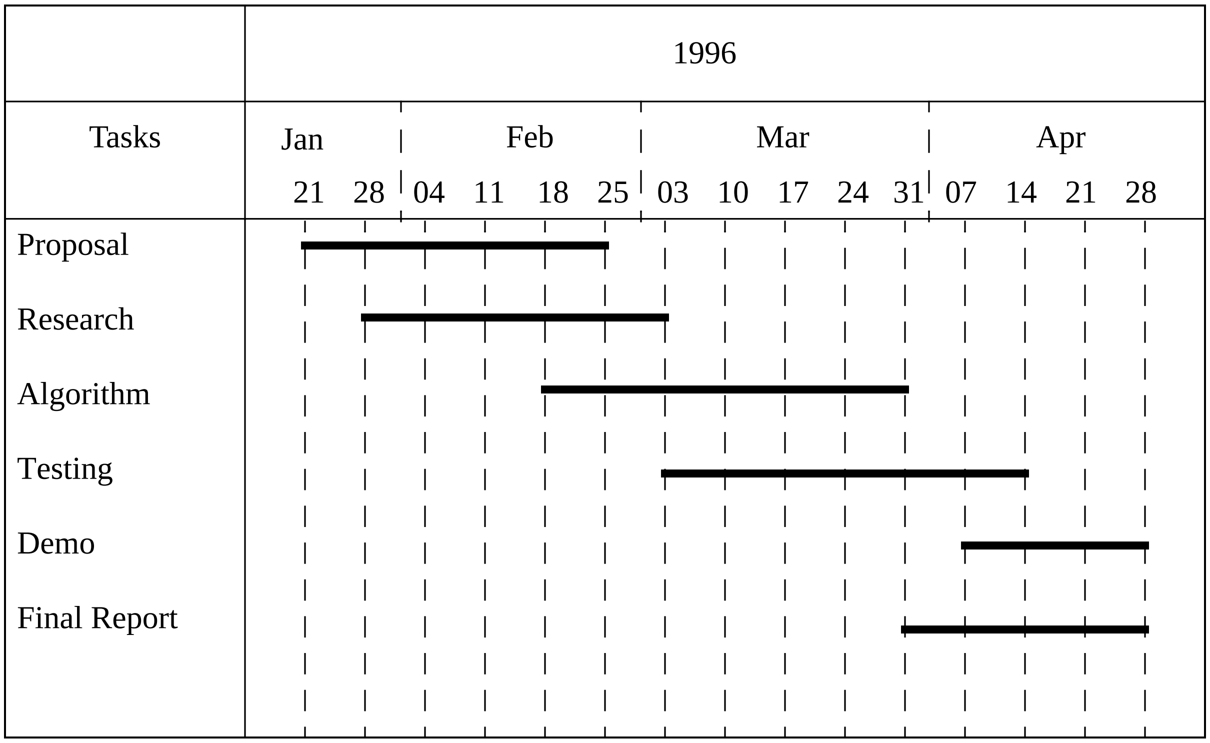


Figure 1. A time-line displaying the projected schedule for the front end of a speech recognition system.

VI. REFERENCES

1 Deller JR, Proakis JG, Hansen JH. Discrete-Time Processing of Signals. Macmillan Publishing Co, New York, NY, 1993.

2 Picone, Joseph. "Continuous Speech Recognition Using Hidden Markov Models," IEEE Acoustics, Speech, and Signal Processing, 1990.

3 Reddy, D Raj. Speech Recognition. Academic Press, New York, Sanfransisco, London, 1975.

4 Haton, Jean-Paul. Automatic Speech Analysis and Recognition. D Reidel Publishing Co, Dordrecht, Holland, 1982.

5 Proakis JG, Manolakis DG. Digital Signal Processing. Second Edition, Macmillan Publishing Co, New York, 1988.

proposal for

Front End Modeling with Special Emphasis on FFTs, LPC, and Feature Selection

submitted to fulfill the semester project requirement for

EE 8993: Fundamentals of Speech Recognition

February 1, 1996

submitted to:

Dr. Joseph Picone

Department of Electrical and Computer Engineering
413 Simrall, Hardy Rd.
Mississippi State University
Box 9571
MS State, MS 39762

submitted by:

L. Webster

Department of Electrical and Computer Engineering
Mississippi State University
Box 9571
Mississippi State, Mississippi 39762
Tel: 601-325-3149
Fax: 601-325-2298
email: law2@ra.msstate.edu

I. ABSTRACT

Communication is a key factor in life. In order to be a productive communicator, speech must be generated and comprehended, by fully understanding the speech signal. Information theory states that speech can be represented in terms of its message content. Another way of describing speech is in terms of an acoustic waveform, the signal relaying the message content. A typical speech recognizer is composed of three main elements: the front-end or acoustic model, search, and language modelling. In this proposal, the first element of a speech recognizer, the front-end model will be discussed with special emphasis given to fast Fourier transform (FFT) based measurements, linear prediction coefficient (LPC) transformations, and feature selection. The over-all purpose of this proposal is to design and implement the specific aspects of the acoustic model mentioned above with the final goal of incorporating them into a speech recognition system.

II. INTRODUCTION

The conversion of an acoustic waveform into a written equivalent of the information transmitted is speech recognition. Speech recognition is very sensitive to the constraints placed on a speaker, the speaking environment, and the context of the information. Future applications for speech recognition systems remains unbounded. From directory assistance in telephone communications to voice communication with computers, the need for sound, robust speech recognition systems is evident.

One of the first steps in speech recognition is the parameterization of a speech signal. Meaningful parameters which simulate human auditory and perceptual systems aid algorithms that model the speech signal. The goal of these algorithms is to maximize the speech recognizer's performance. For this reason, signal modelling is an essential part of a speech recognition system. Signal modelling can best be defined as the means by which sequences of speech signals are adapted to vectors to depict events in a probability space.

Four basic components of signal modelling are spectral shaping, spectral analysis, parameter transformation, and statistical modelling. Spectral shaping includes A/D conversion (from a sound pressure wave to a digital signal) and digital filtering (focusing on key frequency components in the speech signal). Spectral analysis, the topic of this proposal, is formed by six major components: digital filter bank, the Fourier transform filter bank, cepstral coefficients, linear prediction coefficients, linear prediction-derived filter bank amplitudes, and linear prediction-derived cepstral coefficients. Today, both the Fourier transform and linear prediction play highly instrumental roles in various speech processing applications. Parameters transformed from the signal are processed in two ways: differentiation and concatenation. A parameter vector that contains the lowest estimates of the signal is the output of this stage. The final aspect of signal modelling, statistical modelling of the signal parameter, involves forcing a model on the data, training the model, and measuring the quality of the approximation. Because statistical modelling is a fundamental function of a speech recognition system, extremely knowledgeable models are implemented.[1]

III. PROJECT SUMMARY

This project is divided into three main components of an acoustic model. All code will be written in the C++ programming language and conform to the ISIP standards to allow for easier implementation into a full speech recognition system. The three functions of signal modelling to be implemented are: FFT based measurements (including cepstral analysis), LPC transformations, and feature selection.

FFT Based Measurements

When a sequence is periodic with a period of N,

$$\tilde{x}(n) = \tilde{x}(n+N) \qquad -\infty < n < \infty$$

A signal, x(n), can be depicted by a discrete sum. For a periodic sequence, the Fourier series representation is

$$\tilde{x}(k) = \sum_{n=0}^{N-1} \tilde{x}(n) e^{-j2\pi k \frac{n}{N}}$$

$$\tilde{x}(n) = \frac{1}{N} \sum_{k=0}^{N-1} \tilde{x}(k) e^{j2\pi k \frac{n}{N}}$$

The above formulas are an exact formulation of a periodic sequence. For a finite sequence, x(n), which is zero outside [0,N-1], the z-transform can be shown as

$$X(z) = \sum_{n=0}^{N-1} x(n) z^{-n}$$

When x(z) is evaluated at equally spaced points on the unit circle, a periodic sequence, x(n), is obtained.

$$x(n) = \sum_{r=-\infty}^{\infty} x(n+rN)$$

Therefore, we have

$$x\left(e^{j2\pi\frac{k}{N}}\right) = \sum_{n=0}^{N-1} x(n) e^{-j2\pi k \frac{n}{N}} \qquad k = 0, 1, \ldots, N-1$$

The samples are the Fourier coefficients of x(n) in (e), and a sequence (N long) can be represented by a discrete Fourier transform (DFT)

$$X(k) = \sum_{n=0}^{N-1} x(n) e^{-j2\pi k \frac{n}{N}} \qquad k = 0, 1, \ldots, N-1$$

$$x(n) = \frac{1}{N} \sum_{k=0}^{N-1} X(k) e^{j2\pi k \frac{n}{N}} \qquad n = 0, 1, \ldots, N-1$$

The double parenthesis implies the periodicity of the DFT, which has an important effect on the properties of the DFT. One of the most crucial aspects of the DFT is that the N values of X(k) can be computed very efficiently by a set of computational algorithms known as the fast Fourier transform (FFT). DFTs are usually used for the computation of spectral estimates, correlation functions, and for implementing digital filters. [2]

The development of computationally efficient algorithms for the DFT is made possible with a divide and conquer approach that is based on the decomposition of an N point DFT into successively smaller DFTs. This leads to FFT algorithms.[3]

The inverse DFT (IDFT) allows x(n) to be calculated from

$X\left(e^{j2\pi \frac{k}{N}}\right)$ by $x(n) = \frac{1}{N}\left(DFT\left(x^{*} e^{j2\pi \frac{k}{N}}\right)\right)^{*}$ where $*$ is the complex conjugate. This expression can be used to obtain the DFT and the IDFT from a single DFT program. The FFT can be used to implement the DFT of any sequence possessing a length that is a power of 2. If the length fails to be a power of 2, zeros can be added to the sequence.

Using a radix-2 algorithm, the most widely used FFT algorithm, consider N=2 where N=LM (the product of 2 integers). Let M=N/2 and L=2. This splits the N-point data sequence into two N/2 data sequences of $f_1(n)$ and $f_2(n)$, the even and odd samples of x(n).

$$\begin{aligned} f_1(n) &= x(2n) \\ f_2(n) &= x(2n+1) \end{aligned} \qquad n = 0, 1, \ldots, \frac{N}{2} - 1$$

Now, $f_1(n)$ and $f_2(n)$ are found by reducing x(n) by a factor of 2, resulting in the FFT algorithm, a decimation-in-time algorithm.

The N-point DFT can be expressed in terms of the DFTs of the reduced sequences as follows:

$$\begin{aligned} X(k) &= \sum_{n=0}^{N-1} x(n) W_N^{kn} \qquad k = 0, 1, \ldots, N-1 \\ &= \sum_{n\,even} x\langle n\rangle W_N^{kn} + \sum_{n\,odd} x\langle n\rangle W_N^{kn} \\ &= \sum_{m=0}^{\frac{N}{2}-1} x\langle 2m\rangle W_N^{2mk} + \sum_{m=0}^{\frac{N}{2}-1} x\langle 2m+1\rangle W_N^{k2m+1} \end{aligned}$$

where $W_N = e^{-j2\frac{\pi}{N}}$ and $W_N^2 = W_{\frac{N}{2}}$. So,

$$\begin{aligned} X(k) &= \sum_{m=0}^{\frac{N}{2}-1} f_1(m) W_{\frac{N}{2}}^{km} + W_N^k \sum_{m=0}^{\frac{N}{2}-1} f_2(m) W^{km}{}_{\frac{N}{2}} \\ &= F_1\langle k\rangle + W_N^k F_2\langle k\rangle \qquad k = 0, 1, \ldots, N-1 \end{aligned}$$

$F_1(k)$ and $F_2(k)$ are the N/2 point DFT sequences of $f_1(m)$ and $f_2(m)$.

Because $F_1(k)$ and $F_2(k)$ are periodic with T=N/2. For this reason,

$$F_1\left(k + \frac{N}{2}\right) = F_1(k)$$

$$F_2\left(k + \frac{N}{2}\right) = F_2(k)$$

$$W_N^{k + \frac{N}{2}} = -W_N^k$$

and

$$X(k) = F_1(k) + W_N^k F_2(k) \qquad k = 0., 1, \ldots, \frac{N}{2} - 1$$

$$X\left(k + \frac{N}{2}\right) = F\ \ (k) - W_N^k F_2(k) \qquad k = 0, 1, \ldots, \frac{N}{2} - 1$$

The reduction of the data may be repeated until the end sequences are one-point sequences. For N=2, reduction can be performed $v=\log_2 N$ times. The number of multiplications is reduced to (N/2)$\log_2$N. Therefore, as the number of points increases, the FFT improves the speed of computations.

A decimation-in-frequency algorithm also exists where M=2 and L=N/2 following the same basic procedures for the decimation-in-time algorithm.

Basically, the radix-2 FFT algorithm takes 2 data points from memory, performs computations, and returns the resulting numbers to memory. This is repeated $(N\log_2 N)/2$ times. The FFT is a useful tool in linear filtering, correlation, and spectrum analysis. [4]

LPC Transformations

Linear prediction (LP) is one of the most powerful speech analysis tools, especially in estimating the basic speech parameters of pitch, formats, spectra, etc. This method offers very good estimates of speech parameters in addition to efficient computation.

The key idea of LP is that a speech sample value can be approximated as a linear combination of past speech samples. By minimizing the sum of the mean-squared error, a unique set of prediction coefficients can be determined. A linear prediction method provides a reliable, accurate, and robust procedure for estimating the parameters of speech.

Various formulations of linear prediction analysis have been:

1. Covariance method.
2. Autocorrelation formulation.
3. Lattice method.
4. Inverse filter formulation.
5. Spectral estimation formulation.
6. Maximum likelihood formulation.
7. Inner product formulation.

Consider the steady-state system function of a digital filter.

$$H(z) = \frac{S(z)}{U(z)}$$

$$= \frac{G}{1 - \sum_{k=1}^{p} a_k z^{-k}}$$

For voiced speech, an impulse train stimulates the system. For unvoiced speech, a random noise sequence is the stimuli. The parameters of this model are: voiced/unvoiced, pitch, gain (G), and the digital filter's coefficients {a_k}. The speech samples s(n) are related to the excitation by

$$s(n) = \sum_{k=1}^{p} a_k s(n-k) + Gu(n)$$

A linear predictor with prediction coefficients a_k has the output

$$\tilde{s}(n) = \sum_{k=1}^{p} a_k s(n-k)$$

The predictor error is

$$e(n) = s(n) - \tilde{s}(n) = s(n) - \sum_{k=1}^{p} a_k s(n-k)$$

From e(n), the system transformation function outputs the prediction error sequence

$$A(z) = 1 - \sum_{k=1}^{p} a_k z^{-k}$$

If a speech signal obeys A(z), then e(n)=Gu(n). Thus A(z) is an inverse filter for the system H(z).

$$H(z) = \frac{G}{A(z)}$$

The basic problem now is to find a set of predictor coefficients a_k from the speech signal to minimize the error. The coefficients must be estimated from short segments of speech. A set of coefficients that minimize the mean-squared prediction error over these short speech segments must be found. The resulting parameters are then assumed to be the parameters of H(z).

The short-time average prediction error is

$$E_n = \sum_m (s_n(m) - \tilde{s}_n(m))^2$$

$$= \sum_m \langle s_n\langle m\rangle - \sum_{k=1}^{p} a_k s_n\langle m-k\rangle\rangle^2$$

$s_n(m)$ is the speech segment chosen in the neighborhood of sample n. We can find values of a_k that minimize En by

$$\frac{\partial}{\partial a_i} En = 0 \qquad i = 1, 2, \ldots, p$$

which results in

$$\sum_m s_n(m-i)s_n(m) = \sum_{k=1}^{p} \hat{a}_k \sum_m s_n(m-i)s_n(m-k) \qquad 1 \le i \le p$$

$\hat{a}_k$ are values of a_k that minimize En.

Defining

$$\phi_n(i,k) = \sum_m s_n(m-i)s_n(m-k)$$

we find that

$$\sum_{k=1}^{p} a_k \phi(i,k) = \phi_n(i,0) \qquad i = 1, 2, \ldots, p$$

This set of p equations with p unknowns can be solved for the unknown predicator coefficients a_k. The minimum mean squared prediction error is

$$En = \sum s_n^2(m) - \sum_{k=1}^{p} a_k \sum_m s_n(m)s_n(m-k)$$

and

$$En = \phi_n(0,0) - \sum_{k=1}^{p} a_k \phi_n(o,k)$$

Therefore, the total minimum error consists of a fixed component and another component that is depends on the predictor coefficients.

To solve for the best predictor coefficients, the quantities $\phi_n(i,k)$ for $1 \le i \le p$ and $0 \le k \le p$ must be calculated and solved to give the a_k 's.[4]

Feature Selection

Feature selection or feature extraction and their evaluation in terms of classification performance are two tasks that are pretty much inseparable because performance evaluation is fused into the search for good features.

Features should be evaluated in terms of minimizing the rate of error. Error rate is usually a very difficult measure to evaluate, so other techniques must be implemented. Most of the techniques in feature selection involve attempting to measure the separation of classes represented by features.

Distance measures, namely Euclidean, are the easiest techniques for measuring class separation. Probabilistic measures represent an attempt to capture that information in the evaluation.

To show probabilistic distance, look at the two-class problem where class-conditional probability distance functions for two contrasting features, x and y. Assume that the *a priori* class probabilities are equal P(c=1) = P(c=2). In one case, features characterized by a random variable x are implemented and $f_{\bar{x}|\bar{c}}(x|1)$ and $f_{\bar{x}|\bar{c}}(x|2)$ are separated with respect to the values of the feature. The classes are almost fully separable because of these densities. On the other hand, when y is used, the separation is poor. In this case, $f_{\bar{y}|\bar{c}}(y|1)$ and $f_{\bar{y}|\bar{c}}(y|2)$ are identical, and the classes would be completely inseparable because of this feature. Pure guessing offers the same amount of performance as this feature.

Probabilistic distance measures try to capture the degree of overlap of the class pdf's as a measure of their separability. In general, these measures can be shown as

$$J = \left\{ \int_{-\infty}^{\infty} g\{ f_{\bar{x}|\bar{c}}(x|c), P(\bar{c} = c)\} \, dx \right\} \qquad c = 1, 2, \ldots, k$$

$g(x)$ is some, function and $\int_{-\infty}^{\infty} g(x)dx$ indicates the integral over the N-dimensional hyperplanes of x. Probabilistic distance measures possess the following properties

1. J is non-negative, $J \geq 0$.

2. J is maximum if $f_{\bar{x}|\bar{c}}(x|c) = 0$ when $f_{\bar{x}|\bar{c}}(x|c') \neq 0$ for all $c \neq c'$.

3. J=0 when $f_{\bar{x}|\bar{c}}(x|1) = f_{\bar{x}|\bar{c}}(x|2) = \ldots = f_{\bar{x}|\bar{c}}(x|k)$.

An alternative method for finding the pdf overlap is probabilistic dependence measures which indicate how strongly the feature outcomes are dependent on the class they are associated with. In an extreme cases when the features are independent of the class they are in, the class conditional pdf's are identical to the "mixture" pdf

$$f_{\bar{x}|\bar{c}}(x|c) = f_{\bar{x}}(x)$$

for all c.

Conversely, when the features depend strongly on their class association, $f_{\bar{x}|\bar{c}}(x|c)$ is expected to be different from the mixture pdf. Therefore, a good indicator of the effectiveness of features at separating the classes is given by probabilistic dependence measures which show the difference between the class conditional pdf's and the mixture pdf. These measures follow the same properties noted above for the probability distance measures and form

$$J = \left\{ \int_{-\infty}^{\infty} g\{ f_{\bar{x}|\bar{c}}(x|c),\ f_{\bar{x}}(x), P(\bar{c} = c)\}\, dx \right\} \qquad c = 1, 2, \ldots, k$$ [5].

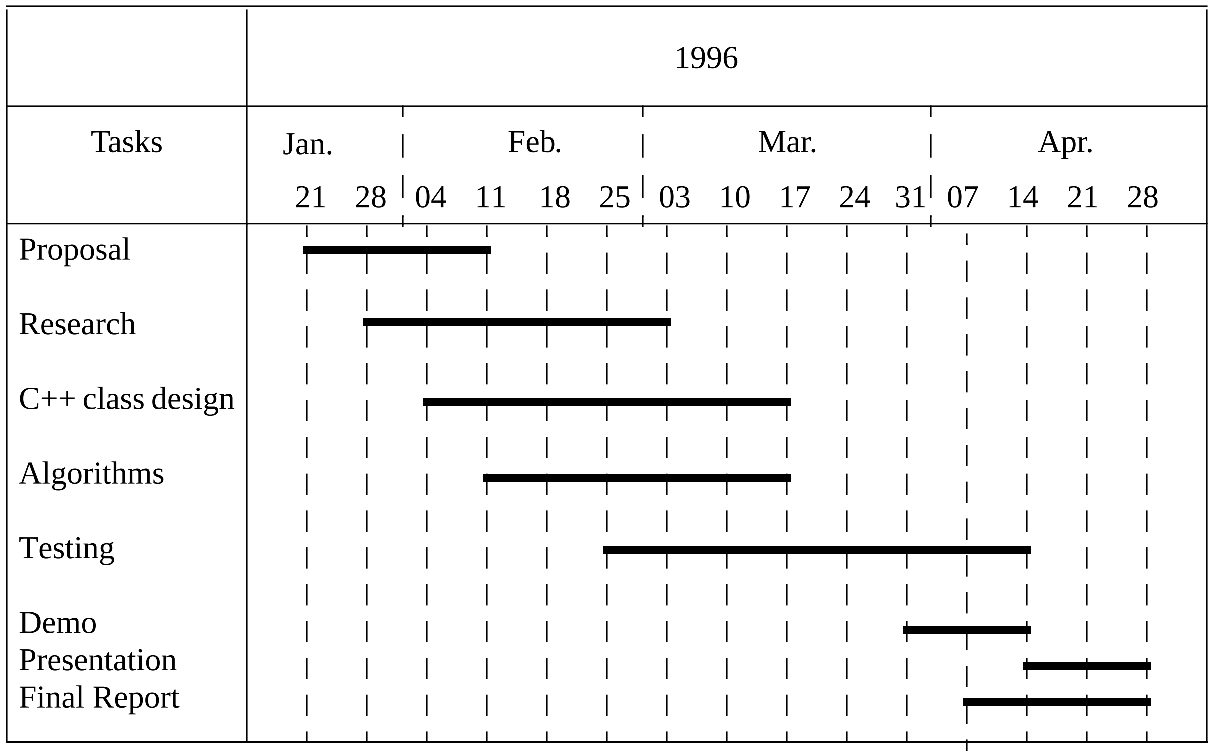
IV. EVALUATION

All three aspects (FFT measurements, linear prediction, and feature selection) will be implemented using the C++ programming language conforming to ISIP standards. The first code to be written and tested will be the LPC code. To test this code, a sinusoidal signal will be used as the input. Once the code has computed the coefficients they will be compared to hand calculated results. After deducing that the LPC phase is operational, the next aspect to be implemented will involve the FFTs. Finally, feature selection will be studied and evaluated.

After all three front end models are correctly configured, each part will be integrated into a speech recognition system that includes more front end modelling, search, and language modelling.

The corpus that will be used in the speech recognition system is quite extensive. Several databases exist today. One such database is the JEIDA database. The JEIDA database consists of isolated digits, four digit strings, etc. that are spoken in Japanese. Other known databases may also be used in the testing of the speech recognizer.

V. SCHEDULE



VI. REFERENCES

1. Picone, J., "Signal Modeling Techniques in Speech Recognition," Proceedings of the IEEE, vol. 81, no. 9, pp. 1215-1246, Sept. 1993.
2. Rabiner, L.R. and R.W. Schafer, *Digital Processing of Speech Signals,* Prentice-Hall, Inc., 1978.
3. Proakis, J. G. and D. G. Mandakis, *Digital Signal Processing: Principles, Algorithms, and Applications*, Macmillian Publishing, 1988.
4. Markel, J.D. and A.H. Gray, Jr., *Linear Prediction of Speech*, Springer-Verlag Publishers, 1982.
5. Deller, J. R., Jr., J. G. Proakis, and J. H.L. Hansen, *Discrete-Time Processing of Speech Signals,* Macmillian Publishing, 1993.
6. Flanagan, J. L. and L.R. Rabiner, *Speech Synthesis*, Dowden Hutchinson and Ross, Inc., 1973.
7. http://www.uninova.pt/~tr/home/tooldiag.html
8. http://www.speech.su.oz.au/comp.speech/Section6/Q6.5.html
9. ftp://ftp.cs.cmu.edu/project/fgdata/speech-compression/LPC/
10. http://www.lhs.com/

proposal for

Implementation of Statistical Modeling Techniques and Channel Adaptation Techniques

submitted to fulfill the semester project requirement for

EE 8993: Fundamentals of Speech Recognition

February 1, 1996

submitted to:

Dr. Joseph Picone

Department of Electrical and Computer Engineering
413 Simrall, Hardy Rd.
Mississippi State University
Box 9571
MS State, MS 39762

submitted by:

R. Seelam

Department of Electrical and Computer Engineering
Mississippi State University
Box 9571
Mississippi State, Mississippi 39762
Tel: 601-325-3149
Fax: 601-325-2298
email: seelam@isip.msstate.edu

ABSTRACT

Implementation of various Statistical Modeling Techniques is necessary for the building of a Speech Recogniser. Statistical Modeling is done to learn the nature of the multi-variate random process generating the signal parameters. In this direction, pre-whitening transformations will be performed on the parameters to eliminate redundancy and to make the analysis easier.

The transformations will be performed on the input vector to produce an uncorrelated Gaussian random vector, containing only "information-bearing" parameters. For some algorithmically complex computations such as the computation of the covariance matrix, existing software will be used. Evaluation will be done by comparing the performance of the software on different classes of data.

Channel adaptation techniques will be implemented so as to make the parameters robust to changes in the acoustical environment. For this purpose, two particularly simple, but effective algorithms, Cepstral Mean Normalisation/Subtraction and RASTA have been chosen. The Codeword-Dependent Cepstral Normalisation method will also be studied. Evaluation will be done by comparing the performance of my software with existing public-domain software.

I. INTRODUCTION

1) *Prewhitening Transformations:* In signal analysis, it is often required to compare signal parameters which have completely different numerical scales, and if a simple operator such as a Euclidean distance is used to make this comparison, the result is likely to be dominated by the terms with large amplitudes and variances, even though the true information may lie in the smaller amplitude parameters[1]. So, appropriate weighting of the parameters has to be done before the comparison. Prewhitening transformations are done on the signal parameters to achieve the same purpose.

Prewhitening of parameters results in the de-correlation of the parameters. Correlation of parameters is undesired for two reasons. Firstly, correlation implies redundancy and so by de-correlation we might be able to achieve some level of compression because the number of "information-bearing" parameters may be less than the number of measured parameters. Secondly, the presence of correlated parameters makes the analysis complex.

2) *Channel Adaptation:* The need for speech recognition systems to be more robust with respect to their acoustic environmental has become more widely appreciated in recent years[4]. Performance of many Automatic Speech Recognition machines, designed to be speaker-independent, has been found to be deteriorating if a different acoustical environment is used to test them[5,6]. So, channel adaptation is being performed to make our system more environmentally robust. This will be achieved by making use of two particularly simple, but effective algorithms, Cepstral Mean Subtraction/Normalization and RASTA[7].

II. PROJECT

Prewhitening Transformations: A linear transformation can be computed that will simultaneously normalize and decorrelate the parameters. Let us define a transformed vector $\bar{y}$ as

$$\bar{y} = \Psi(\bar{v} - \bar{\mu}_v)$$

where $\bar{v}$ denotes the input parameter vector, and $\bar{\mu}_v$ denotes the mean value of the input parameter vector. We define Ψ as a prewhitening transformation[8,9], based on the fact that we desire the output of this transformation to be an uncorrelated Gaussian random vector. To achieve this result, it can be shown that Ψ is given by

$$\Psi = \Lambda^{-1/2}\Phi$$

where Λ denotes a diagonal matrix of eigenvalues, and Φ denotes a matrix of eigenvectors of the covariance matrix of $\bar{v}$. The eigen value and eigenvectors can be shown to satisfy the following relation:

$$C_v = \Phi\Lambda\Phi^{\dagger}$$

where C_v is the covariance matrix for v. Each element in C_v, $C_v(i,j)$, can be computed as follows:

$$C_v(i, j) = \frac{1}{N_f} \sum_{m=0}^{N_{ff}-1} (v_m(i) - \mu_v(j))(v_m(j) - \mu_v(j))$$

Even though this computation is algorithmically complex, existing software[10] will be used to achieve this.If the parameters are uncorrelated, then the covariance matrix reduces to a diagonal matrix. In this case, the transformation simplifies to a diagonal matrix with $1/\sigma_{v(i)}$'s as the diagonal components where $\sigma_{v(i)}$ is the standard deviation of the *i*th component of the parameter vector $\bar{v}$. This is readily recognized as normalizing the parameters by their standard deviations, thereby making each parameter count equally in the calculations. These so-called "variance-weighted cepstral coefficients" are very popular in speech recognition systems today[1].

Channel Adaptation: The two methods, Cepstral Mean Normalization and RASTA have been chosen due to their simplicity and sufficient effectiveness.

Cepstral mean normalization is a simple technique that compensates for channel mis-matches[13] (mainly due to microphone/speaker variability). The normalized and unnormalized cepstral vectors are related as follows:

$$c'_i(t) = c_i(t) - m_i$$

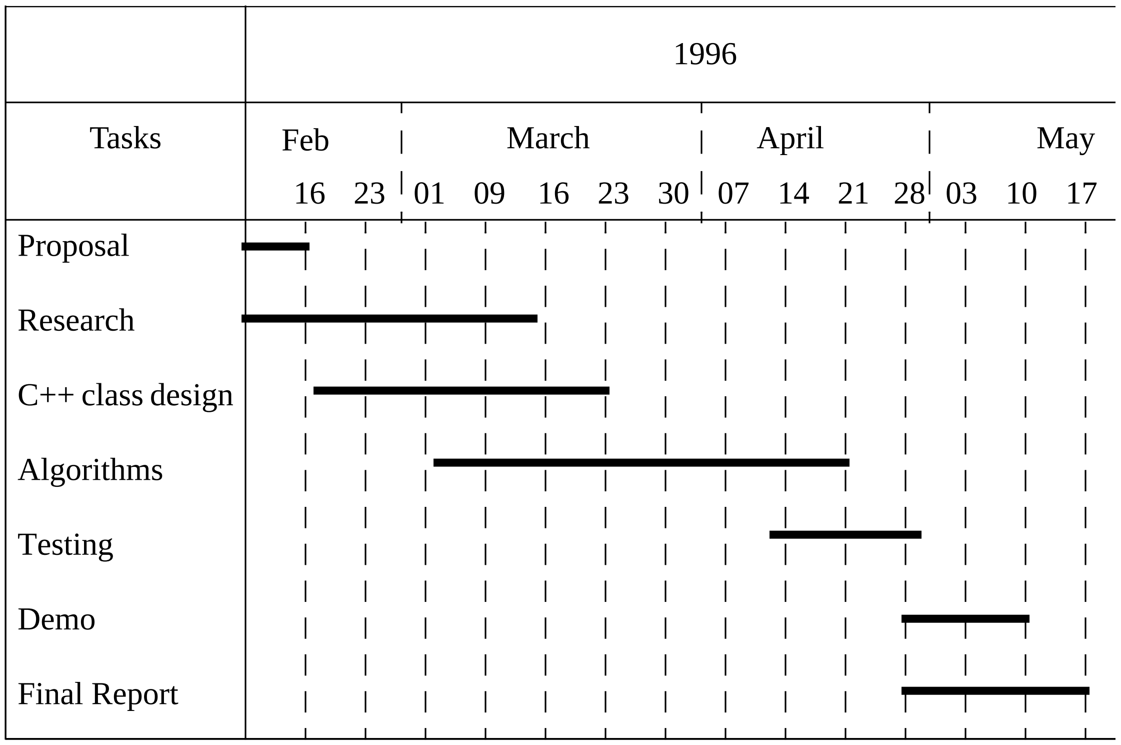
where $m_0 = max_t c_0(t)$, and m_i is the mean value of $c_i(t)$ over an utterance. If we approximate m_i by M_i, the average value of m_i over the training data, we have

$$c'_i(t) \approx c_i(t) - M_i$$

The means and variances of the normalized and unnormalized systems can thus be mapped back and forth using the above equation.

RASTA(RelAtive SpecTrAl) processing[7] was designed to alleviate logarithmic spectral components with rates of change outside the typical rate of change of speech spectral components. By operating in the logarithmic spectral domain, RASTA effectively diminishes spectral components that are additive in the logarithmic spectral domain, in particular the fixed or slowly-changing spectral characteristics of the environment (convolutive in the time domain and therefore additive in the log spectral domain). A modified function will be used for this purpose which is linear-like for small spectral values and approximately logarithmic for large ones. The results of this have been shown to be comparable to training on noisy data[14].

III. SCHEDULE



IV. REFERENCES

[1] J. Picone, “Signal Modeling Techniques in Speech Recognition”, in *Proc. IEEE,* vol. 81, no. 9, pp. 1215-1247, Sep. 1993.
[2] J. Makhoul, S. Raucos, and H. Gish, “Vector Quantization in speech coding”, in *Proc. IEEE*, vol. 73, no. 11, pp. 1551-1588, Nov. 1985.
[3] Y. Linde, A. Buzo, and R.M. Gray, “An algorithm for vector quantizer design”, *IEEE Trans. Commun.,* vol. COM-28, no.1, pp. 84-95, Jan. 1980.
[4] B.H. Juang, “Speech Recognition in Adverse Environments”, *Computer Speech and Language,* Vol 5, No 3, pp 275-294, 1991.
[5] A. Acero, R.M. Stern, “Environmental robustness in Automatic speech recognition”, *Proc ICASSP,* pp 849-852, Apr 1990.
[6] A. Erell, M. Weintraub, “Estimation using Log-spectral-distance criterion for Noise-robust speech recognition”, *Proc ICASSP,* pp 853-856, Apr 1990
[7] H. Hermansky, N. Morgan, A. Bayya, and P. Kohn, “Compensation for the effect of the communication channel in auditory-like analysis of speech (RASTA-PLP)”, *Proc* EUROSPEECH ‘91, pp 1367-1370, Genova, 1991.
[8] E.L. Bocchieri and G.R. Doddington, “Frame specific statistical features for speaker-independent speech recognition”, *IEEE Trans. Acoust., Speech, Signal Processing,* vol. ASSP-34, no.4, pp 755-764, Aug. 1996.
[9] K. Fukunaga, *Introduction to Statistical Pattern Recognition.* New York: Academic Press, 1972
[10] W.H. Press, B.P. Flannery, S. A. Teukolsky, and W.T. Vettering, *Numerical Recipes in C: The Art of Scientific Programming*. New York: Cambridge Univ. Press, 1988.
[11] M.R. Anderberg, *Cluster Analysis for Applications*. New York: Academic Press, 1973.
[12] J.R. Deller, J.G. Proakis, and J.H.L. Hansen, *Discrete Time Processing of Speech signals.* New York: MacMillan, 1993.
[13] R.A. Gopinath, M. Gales, P.S. Gopalakrishnan, S. Balakrishnan-Aiyer and M.A. Picheny, *Proc of ARPA Spoken Language Systems,* 1994
[14] H. Hermansky, N. Morgan, and H.G. Hirsch, “Recognition of speech in additive and convolutional noise based on RASTA spectral processing”, *Proc ICASSP,* vol. 2, pp 83-85, 1993.

proposal for

Implementation of Viterbi Beam Search Algorithm

submitted to fulfill the semester project requirement for

EE 8993: Fundamentals of Speech Recognition

February 1, 1996

submitted to:

Dr. Joseph Picone

Department of Electrical and Computer Engineering
413 Simrall, Hardy Rd.
Mississippi State University
Box 9571
MS State, MS 39762

submitted by:

A. Ganapathiraju

Department of Electrical and Computer Engineering
Mississippi State University
Box 9571
Mississippi State, Mississippi 39762
Tel: 601-325-3149
Fax: 601-325-2298
email: ganapath@isip.msstate.edu

ABSTRACT

Speech Recognition can be treated in a very general sense as a structured search problem. Correct recognition is defined as outputting the most likely word sequence given the language model, the acoustic model and the observed acoustic data. In this project I plan to implement a very commonly used search algorithm, Viterbi Beam Search. For this purpose continuous observation HMMs will be implemented to represent phonemes and will be trained on a large data set. The Viterbi reestimation algorithm will be used for the purpose of training the HMMs. The search algorithm will then be used to output the most likely phoneme sequence that matches the observed acoustic data. The code will be implemented in GNU C++ and the structure will be made very generic so as to allow for using other search algorithms at a later stage. The design will keep in mind the algorithm's integration with various other modules like the language model, and the front-end signal processor, to form a simple continuous speech recognizer. For evaluating the performance of the search engine, results from other public-domain speech recognizers will be compared. One of the main aims of this project will be to study the recognition performance dependence on various aspects of the HMM parameters like the number of states per model.

I. INTRODUCTION

The goal of present day speech recognition technology is to create machines that receive spoken information and act appropriately upon the received information. But, for this to happen the machine should do the recognition perfectly. Present day systems are atleast an order below the performance of the humans. The performance of Isolated word recognizers is getting close to human performance but the recognition of unconstrained continuous speech is still a challenge to researchers. One could differentiate recognition in terms of vocabulary size, isolate-word to continuous speech, speaker dependent to speaker independent recognition etc.

There are two schools of thought in speech recognition, namely the one based on template matching and the other based on stochastic modelling. The early recognizers almost always used template matching techniques in the form of Dynamic Time Warping (DTW). Later many limitations of this approach were identified like, memory requirements for storing templates of all words, and also the massive computational cost of the search process[1,2]. This prompted interest in Hidden Markov Models(HMMs) and Artificial Neural Networks(ANNs) where in a probabilistic modelling of the variability of speech is done.

The process of speech recognition when using the probabilistic modelling techniques is divided into four interrelated processes namely, language modelling, acoustic modelling and finally the search process. Figure 1. describes this process schematically. Talking in a probabilistic sense we can define the recognition problem as '**Given the acoustic sequence Y, we have to derive a word sequence W* such that P (W* | Y) = max P (W | Y) (maximizing over W) where W represents any word in the vocabulary**'. We can restate this problem as the following equation

$$P(\langle W|Y\rangle) = \frac{P(W) \times P(\langle Y|W\rangle)}{P(Y)}$$

[4]. P(W) is the a priori probability of the word sequence and is specified ny the word model, P (Y| W) is computed from the acoustic model and will be estimated by the HMMs, and P(Y) is the probability of the acoustic sequence. The search process finally boils down to finding P (W|Y).

HMMs are characterized by N, the number of states in the model, the transition matrix A, the observation symbol probability distribution and the initial state distribution. The models have to be trained to get an optimized set of the quantities mentioned earlier. HMMs could be used to model words as such or to model sub-word units like phonemes or phones. Once the training is accomplished using a number of algorithms, Baum- Welch algorithm being the most common one, the search process can be solved.

Search can be accomplished in many ways. If a direct method of searching all possible paths is used the computation becomes unwieldy. Thus all CSR search algorithms being used presently represent a restructuring of the direct search by adding. Efficient search algorithms often incorporate most of the following techniques.

● **Optimized Hypothesis Generation:** Here merging of common partial hypotheses is done which is the basis of the tree search.

● **Search Reduction:** Here inferior hypotheses whose partial evaluation is worse than an already completed evaluation or it is worse than some heuristically determined threshold**.**

In this project I propose to implement two search procedures, namely the **Viterbi Beam Search** and the **Fast Search**. The Viterbi beam search is one of the most efficient search algorithms and finds a place in most of the CSRs of today. It gains its advantage from the dynamic programming techniques which reduce the problem size significantly. The fast search algorithm was first implemented my the IBM speech group. This algorithm is similar to the Viterbi Beam Search except that, for pruning and size reduction of the problem a different criterion is used.

II. CREATING & TRAINING THE HMMs:

HMMs form the core of the recognition process.They provide with statistical tools to deal with the recognition problem. There are Discrete Observation HMMs and Continuous Observation HMMs. In my project I propose to use the Continuous Observation HMMs. In this case the formal description of the HMM contains a multivariate pdf characterizing the distribution of the obsevations within each state[5]. The underlying mean of the feature distribution represents the actual value of the feature vector at that state. The continuous type HMMs are being used in most recent systems simply because of the fact that we can avoid accumulating errors form the VQ process when discrete HMMs are used. I propose to use a left-right model.

Once the HMMs are initialized, a known input will be given to the HMMs for reestimation of the parameters of the HMMs to closely represent the actual spoken sub-word. This is the training process of the HMMs. There are two major training algorithms used for training the HMMs. The Viterbi training algorithm and the Baum-Welch(F-B algorithm) algorithm[7]. Both the algorithms have been found to give comparative performance. However, the Viterbi approach has been found to be more computationally efficient. I propose to implement the Viterbi reestimation algorithm. Recognition performance is very sensitive to the initial conditions. Good seed models are needed for good performance. Models using simple parameter sets are trained and then successively refined from the previous stage models. The HMM system will not be built from scratch, rather an initial single covariance matrix is used for all the states. Later iteration will optimize the parameters of each state. Often an initial observation turns out to be manually excising data from a typical pronunciation of the phoneme.

Once the HMMs have been trained, we go into the recognition phase of the problem. This process is nothing but a search for the best sequence of HMM states which could have generated the observation sequence.

III. VITERBI BEAM SEARCH

The search procedure is based on finding the probability that a given HMM could generate the given observation using the best possible sequence of states. A beam consists of all paths at time t whose scores fall within a specific range of the best hypothesis.The beam search could be carried out in a depth-first or a breadth-first approach. The Viterbi beam search follows the breadth-first approach. The Viterbi Beam Search can be defined as follows by a pseudo code.

1. Create N heaps B for the dynamic state beams. N is the length of the acoustic observation

sequence.

```
2. Set t = 0; Add the initial state to B[0] with probability 1.
3. Clear B[t + 1].
4. For each s in B[t]
        1. for each transition from state specified by S
            a) Compute the transition probability
            b) If the destination state is not in B[t+1] then add it to B[t + 1] with its score and
               the back-pointer set to s.
            c) If the destination state is already in B[t + 1] then check if the transition is better. If
               it is, then update the state with the new score and back-pointer to s.
5. If t = N, backtrack and terminate.
6. Find the best state in B[t + 1] and disregard words that are some threshold worse than the best
   state.
7. t = t + 1 and go to step 3.
```

The advantages of the Viterbi Beam Search are its computational efficiency and the well developed knowledge of the data structures which go into this algorithm. There are a number of issues to be kept in mind to reduce the complexity of this search. Processing by iterating over active hypotheses is very useful when a highly constrained grammars are used. With less constrained grammars with all states likely to be active, a state oriented approach is more efficient[8]. Because this process is time synchronous need for normalization before pruning is avoided.

IV. IMPLEMENTATION

I propose to implement the above algorithm using GNU C++. A very hierarchical class structure will be developed. A HMM class will be created with all the necessary member functions which allow both training and the search process. For the search process, classes for the heaps will be created. Each element of the heap will also be a class with the necessary data members on which operations like updates, additions and deletions can be performed. Each element of the heap will have the following information[6]:

a) log-likelihood score b) reference time c) end of word flag

d) end of sentence flag e) back-pointer to parent node and

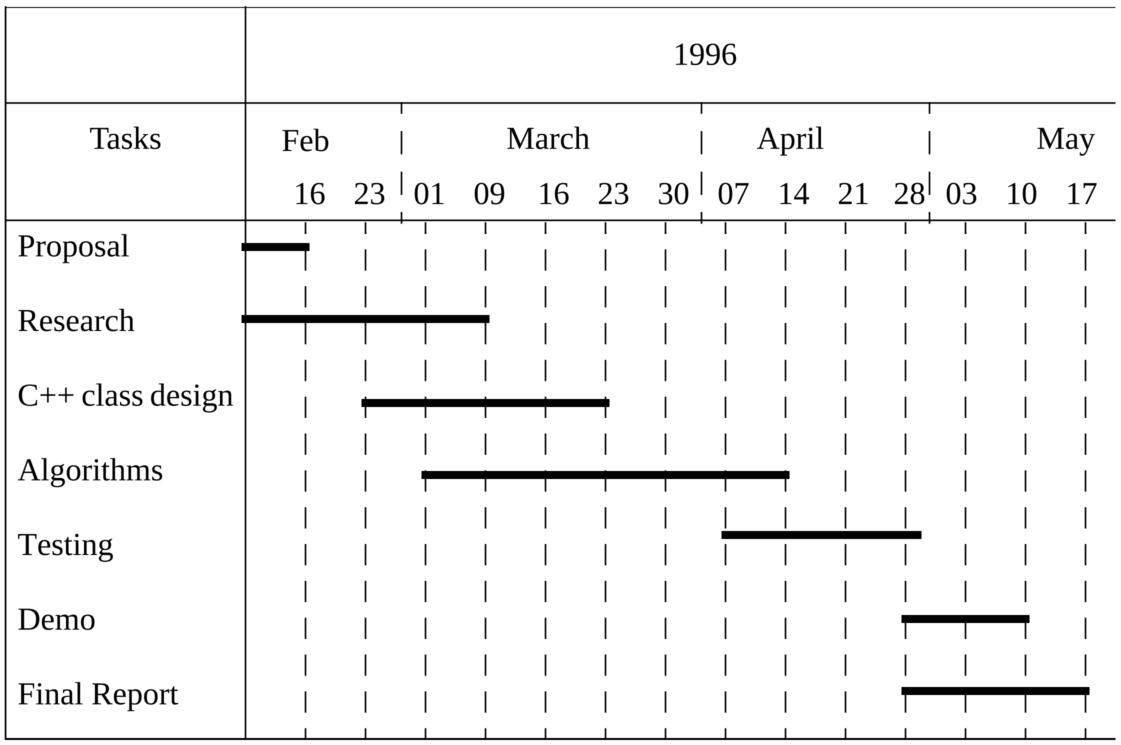
f) a word history i: for path or theory identification.

The structure of the code will be made extensible so that other search algorithms like the N-best Stack-decoding, A* Stack decoder and Tree-Trellis search can also be incorporated into the same framework with minimum redundancy of code.

V. SUMMARY

This project is aimed at understanding the theory and implementation of HMMs and a commonly used Search algorithm. This implementation forms a very important part of a speech recognizer which is planned to be operational by the end of the semester. My initial goal is the proper functioning of the basic structure of the code so that improvements and modifications to the algorithms can be made at a later time. Importance has to be given to the class structure so that this piece of code can be integrated without much difficulty to the other components of the speech recognizer, language model, front-end signal processor etc. Extensive testing on real data has to be done before conclusions on the performance of the algorithms can be drawn. Performance with varying number of states per symbol and varying number of mixture densities per state will be compared.

VI. SCHEDULE



VII. REFERENCES

1 Deller J. R. et.al.," *Discrete Time Processing of Speech Signals*" Macmillan Publishing, 1993, New York.
2 L. R. Rabiner, B. H. Juang, "*Fundamentals of Speech Recognition.*" New Jersey: Prentice

Hall, 1993.

3 Rabiner L.R.," A Tutorial on Hidden Markov Models and Selected Applications in Speech Recognition," *Proceedings of the IEEE*,vol. 77, no. 2, pp. 257-286, Feb. 1989.

4 K. F. Lee, "Automatic speech recognition: the development of the SPHINX system." Boston: Kluwer Academic Publishers, 1989.

5 Picone J.W., "Signal Modelling Techniques in Speech Recognition," *Proceedings of the IEEE*, vol. 81, no. 9, pp. 1215 - 1247, Sept. 1993.

6 Paul, D. B.," An Efficient A* Stack Decoder Algorithm for Continuous Speech Recognition with a Stochastic Language Model," *ICASSP 1992*, vol. 1,pp. 25 - 28.1992.

7 Code of "Hidden Markov Model for Automatic Speech Recognition", Cambridge University Speech Group,U.K.

8 Picone J.W.," Continuous Speech Recognition Using Hidden Markov Models," *IEEE ASSP Magazine*, pp. 26-41, July 1990.

DEPARTMENT OF ELECTRICAL AND COMPUTER ENGINEERING

proposal for

Efficient Search Algorithms for Large Vocabulary Continuous Speech Recognition

submitted to fulfill the semester project requirement for

EE 8993: Fundamentals of Speech Recognition

February 1, 1996

submitted to:

Dr. Joseph Picone

Department of Electrical and Computer Engineering
413 Simrall, Hardy Rd.
Mississippi State University
Box 9571
MS State, MS 39762

submitted by:

N. Deshmukh

Department of Electrical and Computer Engineering
Mississippi State University
Box 9571
Mississippi State, Mississippi 39762
Tel: 601-325-3149
Fax: 601-325-2298
email: deshmukh@isip.msstate.edu

TABLE OF CONTENTS

I. ABSTRACT

Automatic speaker-independent speech recognition has made significant progress from the days of isolated word recognition. Today large-vocabulary continuous speech recognition (LVCSR) over complex domains such as news broadcasts and telephone conversations is a reality. A major component of this success is the result of recent advances in search techniques that support efficient, sub-optimal decoding over large search spaces. These evaluation strategies are capable of dynamically integrating information from a number of diverse knowledge sources to determine the correct word hypothesis.

In this project we propose to implement two major classes of such decoding algorithms viz. multipass N-best search and search using models with a weighted mixture of Gaussian probabilities as density functions. While our implementation of the algorithms will allow it to function as a standalone decoding engine, our final objective is to integrate this project with other software modules implementing a language model and a speech signal-processing front-end to build a complete LVCSR system.

The performance of this LVCSR system will be evaluated on speech data in the public domain and compared with that of other recognizers as a benchmark. We will also evaluate the search algorithm modules in isolation using statistical measures. The final software will be placed in the public domain at the Institute of Signal and Information Processing (ISIP).

II. INTRODUCTION

The problem of continuous speech recognition can be described as constructing an efficient mechanism to automatically perform accurate speech-to-text transcription. A statistical approach is most suitable for this task given the random nature of human speech and the characteristics of the speakers. If a word sequence W is spoken and if A is the acoustic evidence provided to the system to identify it, then the recognizer should rule in favor of a word string $\hat{W}$ that maximizes the probability of W being spoken given A was heard:

$$p(\hat{W}/A) = \max_{W} p(W/A) \tag{1}$$

However, since the *a posteriori* probabilities $p(W/A)$ are not available, using Bayes' expansion we can convert the problem into a more manageable form:

$$\hat{W} = \arg\max_{W} p(A/W)p(W) \tag{2}$$

The decoding process integrates the scores on the *statistical acoustic model* $p(A/W)$ and the *language model* $p(W)$ to generate all possible likely word sequences or hypotheses, and then traverses this search space to come up with the (one or more) most likely hypotheses.

II.A. Historical Perspective

The earlier approaches to continuous speech recognition used the technique of Dynamic Time

Warping (DTW) [1]. However, it was found to be impractical even for moderately large-vocabulary tasks as it places very high requirements on both memory and computational capacity for implementation. Moreover, it suffers from problems of robustness across multiple speakers, interpolation of parallel hypotheses and inappropriate modeling of word-durations.

The present-day state of the art recognizers are typically constructed using Hidden Markov Models (HMMs) [2]. The popularity of HMMs lies in the availability of computationally efficient algorithms for both training the system as well as decoding. In the recent past Artificial Neural Networks (ANNs) have emerged as a viable technique for building speech recognition systems [3]. ANNs are inherently nonlinear and therefore can model the nonlinear nature of speech very well. However, so far in practical implementations ANNs have not shown any significant improvement in performance over systems using HMMs. Systems integrating both HMMs and ANNs are also being developed.

II.B. Search Techniques

In continuous speech recognition the likelihood of the observed data is computed by scoring it on all the feature models. The search paradigm then chooses a speech pattern with the highest likelihood. The number of possible hypotheses grows exponentially with the number of feature HMMs and imposes heavy constraints on the computation and storage requirements. Therefore intuitively obvious techniques such as an exhaustive search are not at all practicable and strategies that save on computation by modifying the search space are vital. These may sometimes cause the system to make suboptimal choices but these are known not to significantly affect the accuracy of recognition. We enumerate a few popular search techniques here:

- Viterbi beam search: This is a breadth-first algorithm that performs a frame-by-frame pruning of hypotheses [4].
- A* or Stack-decoding search: This is a depth-first technique that maintains an ordered stack of the most likely hypotheses [5].
- N-best search: The optimal N-best algorithm is conceptually similar to the Viterbi beam search, except that it maintains all hypotheses within the specified beam width and at every frame allows the top N hypotheses to be retained for further processing [6].

We will discuss N-best search in more detail next.

III. MULTIPASS N-BEST SEARCH

The generalized N-best search has the advantage of using only a subset of the available information to reduce the search space. Knowledge sources that provide more constraints on the system at a lesser cost are used to guide the initial search to generate the list of top N hypotheses. These are then re-scored using other, more expensive information sources to yield the best hypothesis. However, in its pure form it has the drawback of being partial to shorter hypotheses. A modifications that alleviates this problem is the multipass N-best search algorithm [7].

III.A. Forward-Backward Search

The most common form of multipass N-best search uses an approximate time-synchronous search

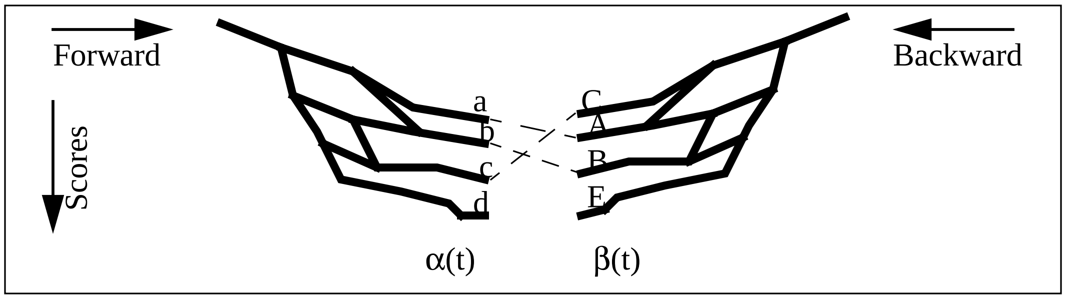


Figure 1: Forward-backward search. Forward and backward scores for the same state and frame are combined to predict final score for each hypothesis [7].

in the forward direction to facilitate a more complex and expensive search in the backward direction [7]. This generally results in speeding up the search process on the backward pass as the number of hypotheses to be explored is limited by the forward coarse search.

A simplified acoustic model (e.g. one using discrete-density HMMs) is used to perform a fast and efficient forward-pass search in which the scores of all partial hypotheses (i.e. word endings with scores higher than a pruning threshold) are saved at each frame. Then using only these words but a detailed (e.g. continuous density HMMs) model, a beam search is performed in the reverse direction to generate the list of N-best hypotheses. The backward search scores high on a hypothesis at a frame only if there also exists a good forward path score leading to that word-ending at that time.

Figure 1 illustrates the search algorithm. Similar to the Baum-Welch training algorithm we combine the scores on the forward and backward passes to compute the overall score at each state s of the HMM at time t. Thus

$$\gamma_t(s) = \frac{\alpha_t(s)\beta_t(s)}{\alpha_T} \tag{3}$$

where α and β are the scores on the forward and backward search passes. The N-best sentences thus obtained are rescored using more sophisticated acoustic and language models to obtain the best sentence hypothesis.

IV. MIXTURE DENSITY MODELS

A continuous-density HMM is the basic building block of our recognition system. It can be formally represented as a four-tuple as described in Equation (4) below.

$$HMM \cong \{S, \pi(1), \mathbf{A}, \{f_{\underline{y}/\underline{x}}(\zeta/i), 1 \le i \le S\}\} \tag{4}$$

where S is the set of states, **A** the matrix of state transition probabilities, $\pi(1)$ the set of initial probabilities for each state and $\{f_{\underline{y}/\underline{x}}(\zeta/i), 1 \le i \le S\}$ the multivariate probability distribution function (pdf) characterizing the observations within each state. Typically this is taken to be a simple Gaussian distribution function.

However, a Gaussian density function quite often fails to be a good approximation of the real distribution of the feature vector. Since the observation pdf should be able to model the acoustic features of the observed signal accurately to achieve good performance, an alternative is to use a mixture i.e. weighted sum of Gaussian distributions as the pdf. In this case we get

$$f_{\underline{y}/\underline{x}}(\zeta / i) = \sum_{m=1}^{M} c_{im} \aleph(\zeta; \mu_{im}, C_{im}) \tag{5}$$

In order for this to be a properly normalized pdf we need the constraints that the weights be non-negative and that

$$\sum_{m=1}^{M} c_{im} = 1 \qquad ;1 \le i \le S \tag{6}$$

We will implement a Baum-Welch kind of forward-backward training algorithm to re-estimate the weights and the component pdfs. The details of the algorithm and references can be found in [1].

V. EVALUATION

We propose to evaluate our search engine as part of the complete LVCSR system which is the cumulative outcome of this course project. The recognition system is being designed to accept speech input in the SPHERE format and will be trained and evaluated on speech data available at ISIP and in the public domain [8]. Word and sentence error rates as well as the time taken to perform recognition as a function of vocabulary size will be the criteria used for benchmarking. Important considerations in the mixture-distributions case will be the number of mixtures required per state of the HMMs and the improvement in performance over models constructed of single Gaussian pdf HMMs compared to the cost of additional computation for re-estimation of the mixture parameters. A comparison with known performance of other systems on similar tasks will also be a measure of success of this implementation.

Before integrating our modules in the overall system we will also confirm its functionality by testing on artificially created non-speech sample inputs, and measuring the statistical deviation from the expected results.

VI. IMPLEMENTATION ISSUES

We are committed to building object-oriented modular software that can be integrated into any related application. As per this philosophy the implementation of the above algorithms will be done in C++ and the final code will be placed in public domain through the resources of the Institute for Signal and Information Processing (ISIP) at Mississippi State University.

VII. SCHEDULE

A brief outline of the planned course of events in this project is summarized in Figure 2.

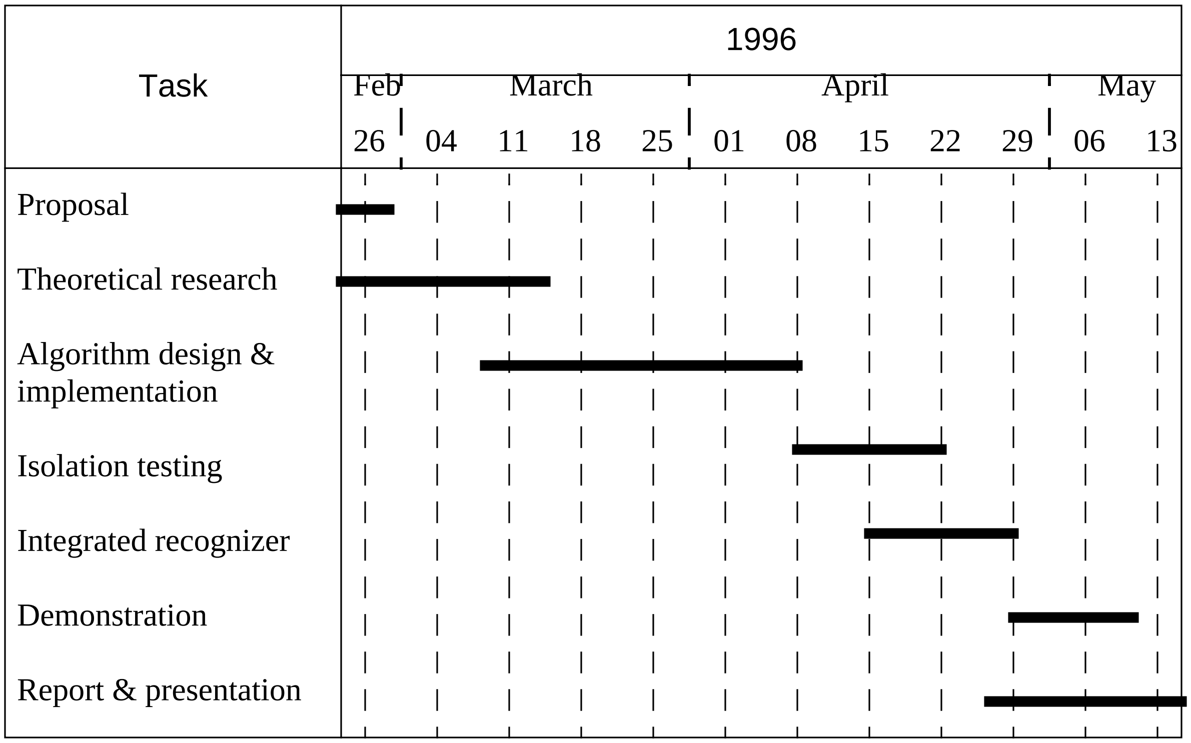


Figure 2: A summary timetable of the planned course of action for the project

VIII. REFERENCES

1. Deller J.R.,Proakis J.G. and Hansen J.H.L., *Discrete-Time Processing of Speech Signals*, Macmillan Publishing, New York, 1993.
2. Rabiner L.R., "A Tutorial on Hidden Markov Models and Selected Applications in Speech Recognition", *in Proceedings of IEEE*, Vol. 77, No. 2, pp. 257-285, 1989.
3. Lippman R.P. and Gold B., "Neural-Net Classifiers Useful for Speech Recognition", i*n Proceedings of the 1st IEEE International Conference on Neural Networks*, Vol. IV, pp. 417-425, San Diego, CA, 1987.
4. Viterbi A.J., "Error Bounds for Convolutional Codes and an Asymptotically Optimal Decoding Algorithm", *in IEEE Transactions on Information Theory*, Vol. IT-13, pp. 260-269, April 1967.
5. Paul D.B., "An Efficient A* Stack Decoder Algorithm for Continuous Speech Recognition with a Stochastic Language Model", *in Proceedings of the IEEE International Conference on Acoustics, Speech and Signal Processing*, pp. 405-409, 1992.
6. Chow Y.L. and Schwartz R.M., "The N-Best algorithm: An Efficient Procedure for Finding Top N Sentence Hypotheses", *in Proceedings of DARPA Speech and Natural Language Workshop*, pp. 199-202, October 1989.
7. Schwartz R.M. and Austin S.,"Efficient, High-Performance Algorithms for N-Best Search", *in Proceedings of the DARPA Speech and Natural Language Workshop*, pp. 6-11, June 1990.
8. The WWW Resources.

proposal for

Language Modeling and Grammar Construction for an HMM Continuous Speech Recognition System

submitted to fulfill the semester project requirement for

EE 8993: Fundamentals of Speech Recognition

February 1, 1996

submitted to:

Dr. Joseph Picone

Department of Electrical and Computer Engineering
413 Simrall, Hardy Rd.
Mississippi State University
Box 9571
MS State, MS 39762

submitted by:

O. LaGarde

Department of Computer Science
Mississippi State University
Box 9571
Mississippi State, Mississippi 39762
Tel: 601-325-3149
Fax: 601-325-2298
email: oml1@ra.msstate.edu

I. ABSTRACT

This project will consist of the implementation of Language Model (LM) objects for the construction of regular stochastic grammars based on Bigrams and Ngrams for both words in a training text and for phones in phonetic series equivalents for those words. An externally produced word-to-phone dictionary compliant with the Worldbet symbol set will be acquired as a supporting resource. A method of deriving Ngrams as the joint product of weighted Bigrams will be defined and investigated.

Implementation of grammar construction and polling objects will target a Continuous Speech Recognition (CSR) system's search engine as the principle user; the objects will provide query response services for probabilities of current and proposed states in the search engine's domain as well as sequences of hypothesized next-states. The models will serve primarily to assist in formation of sentences from hypothesized word sequences through the implementation of a token-based grammar and Unigram/Bigram/Ngram generation scheme. Although the current CSR organization does not require phonetic-level support, such support will implicitly be provided in the token-based model organization. An inherent secondary goal of the project is the creation of a set of robust LM objects for use in CSR experimentation. As such, other models such as Context Free and Case Based Grammars will be investigated, but since this project will be conducted in parallel with projects developing other aspects of the target CSR system, priority for model implementation will be defined exclusively by the needs of the remaining project teams.

The LM will be tested using input perplexity as a benchmark and both alternate Ngram generation methods and independently produced models providing similar services as measures of performance. Deliverables will include object interface and implementation specifications, results of Ngram generation methodology experimentation, and a set of C++ coded language model construction and request servicing objects.

II. INTRODUCTION

This project will develop a Language Model (LM) as a series of utilities for construction of stochastic grammars [1] for use with a Hidden Markov Model (HMM) in a Continuous Speech Recognition (CSR) system. The utilities will consume a text corpus, generate an N-gram model of that text, and use that model to provide probabilistic measures for hypothesized next states given a sequence of proceeding tokens. Since the recognition system's queries to the LM are expected to require both English text and phonetic sequence formats, construction of a word-phone dictionary for the given input text will also be supported. Inclusion of phonetic sequences also prompts the definition of delimiters. Word-grammars will consider a collection of sentences as either a sequence of tokens including words and sentence break symbols such that possible token series may extend across sentences, or as discrete instances of token series bounded by implicit sentence breaks such that possible token series may not cross sentence boundaries. Phone-grammars will consider words as discrete phone sequences only.

Type and quality of the language model can be crucial to performance of the speech recognition system through a variety of factors. Representation of a priori knowledge in the input domain, required to estimate the existential probability of a given input sequence, is imbedded in the LM alone. Constraints of the model (vocabulary, processing speed, etc.) define the search space volume relative to that domain and, to a fair degree, the functionality of the resulting recognition system. Accuracy of the model in calculation of probabilities defines the maximum rate of convergence toward a desirable solution in that search space [1,2].

This project design assumes an intended use with a HMM CSR system and will therefore be most directly applicable to such an application; the primary project goal is therefore to implement as C++ objects a series of utility modules which in turn will provide LM services for a specific HMM CSR implementation. Of equal importance, however, is the creation of robust LM objects that will aid in speech recognition development and experimentation with other speech recognition packages[3,4].

Topics for investigation in the project center on stochastic LM designs which have been well known for some time [1]. In particular, the plan of attack centers on Bigram, Trigram, Ngram, and Co-occurrence probability measures, though difficulties with higher length Ngrams for large vocabularies[2,5] are expected to limit feasible approaches to Trigram implementations.

III. PROJECT SUMMARY

In general, the goal of this project is to supply a HMM search engine with probabilities of correctness for a specific next state (word) given a specific series of previous states; the search engine can then maximize the probability that a hypothesized text sequence is the correct equivalent for a speech input sequence. This, in turn, maximizes performance of the CSR in terms of correct decoding of a speech input into text output. The LM must therefore provide *a priori* contextual knowledge for the linguistic domain of the speech input through access to a body of text representative of the domain of expected speech[6].

The search engine, in locating the best path, creates several additional requirements for the LM functionality; since the recognition system in continuous, the search engine must incrementally decode speech to text across all input for a given execution. This incremental best-path search will require measures of fitness incremental across the input speech sequence, which will require the LM to supply measures of fitness for possible next-states given relatively short sequences of previous states. The search engine can also be expected to justify the pruning of lower quality paths by testing the fitness of relatively long sequences, which will require the LM to supply measures given longer previous-state sequences. Last, the search engine will, as a rule, deal with both words as text and words as phones; the LM services can be expected to operate with both input formats. These four requirements -- context, short sequence fitness, long sequence fitness, and word/phone support -- form the core of functional requirements for implementation. In terms of tasks performed for the CSR, the functional requirements of the LM can be expressed as the generation of both linguistic and phonetic grammars and the calculation, on a single token and token series basis, of existential probabilities; phonetic grammars will assist in the hypothesis of probable words while linguistic grammars will assist in the hypothesis of probable sentences.

It is important to note that the purpose of providing measures of probability is to guide the HMM search engine in selection of next-states given current and past states; this does not necessarily require probabilities, but a means of *consistent ranking relative to probability.* The CSR system is by definition operating in real time; in general, the faster we can process tasks -- without appreciable loss in global performance -- the better our resulting system.

3.1. Linguistic and Phonetic Grammar Generation

Generation of a grammar for a given text will consist of construction of an extended Word Frequency Vocabulary (WFL) in which will be imbedded values and weighted pointers for Unigram and Bigram measures. The model will then consist of the combination of Vocabulary, Unigram list, Bigram list resources, and application of procedures to generate Ngrams from those resources.

The WFL representation schema will token based; words and sentence terminator symbols will comprise the token alphabet for linguistic grammars while phones will comprise the phonetic grammar alphabet. Although the CSR search engine will employ the Viterbi Beam Search algorithm [8] and is not presently expected to require linguistic model coverage of token sequences across sentences or phonetic model coverage across words, this LM project will implicitly supply the capability by representing sentence breaks as special-case tokens. Run-time configuration for or against such support will be added to the project definition if the need arises.

3.2. Unigrams -- Single Token Measures

The generation of Unigrams -- measures of probability for the existence of single tokens -- is arguably the simplest of tasks. Given a training text, the probability of existence for a given token can be expressed statistically as the number of instances, or frequency, of that token over the number of instances of all tokens *n* in the text, or

$$P(t_i) = \frac{F(t_i)}{\sum_n F(t_n)}$$

The task is further simplified by the existence of the WFL, which contains frequencies of occurrence for each unique token in the training text. Counting function then becomes a division of the sum of all WFL entry frequencies by the frequency of the target entry,

$$P(t_i) = \frac{WFL[i, F]}{\sum_n WFL[n, F]}$$

Unigram queries can be construed to fall into the Ngram query category since an Ngram query can be defined to support token sequences of one. For this project, however, we will maintain queries for single tokens, two-token series, three-token series, and n-token series as separate entities. Unigram queries to the LM will exist only in the form of a query for a P(t_i) for a specific token *i* and result only in the application of the above formula.

3.3. Bigrams, Trigrams, and Ngrams -- Token Series Measures

The generation of Bigrams and Trigrams -- existential probabilities for a two-token series -- is similar to that of Unigram generation in that the basic task still reduces to a counting problem, though in this case the counting problem begins to look familiar to anyone acquainted with the fundamental equation of speech recognition and Bayes rule[5,7]. The measure can be statistically expressed as the frequency of the series in the training text -- similar to the calculation of Unigrams, though we cannot use the WFL to reduce the task further reduce the counting task -- by considering the proposed Bigram itself as a token.

Since the Bigram list is expected to be relatively small and scanning the training text prior to grammar generation is already required to generate the WFL and Unigram lists, will take the opportunity to include pointers in our WFL indicating the following token and weighted with the frequency of that token pair found in the training text. The probability for a given Bigram or the measure of the existence of token *b* given a proceeding token *a* can then be statistically expressed as the frequency of that token pair β_i over the sum of frequencies of all token pairs *n*,

$$P(\beta_i) = \frac{F(\beta_i)}{\sum_n F(\beta_n)}, \qquad \beta_i = \{a, b\}$$

Two factors prohibit the calculation of Trigrams and Ngrams in similar fashion -- processing time and storage space. Such calculations require further scanning of the training text, which adds disk I/O time to our operational speed, or requires us to store the entire training text in memory, usually impossible given hardware constraints. Furthermore, calculation for an *n*-order series of tokens would require frequency measures for all *n*-ary combinations of members of the vocabulary occurring in the training text; the processing and storage requirements to generate the list of possible series alone is clearly prohibitive, though for texts of lower perplexity, generation of a Trigrams list is often acceptable[4].

Rather than dedicate to the explicit calculation of Trigrams, however, we can note that Ngrams may be calculated as the joint probability of intersecting Ngrams of lower order existing in the target Ngram and recursively define all Ngrams of order greater than two (Trigrams included) as a sequence of overlapping Bigrams. This method does require the existence of a transitive property across probabilities for lower order token series that does not necessarily exist, but current sources of stochastic LM development[1,<3list>] indicate satisfactory results with the application of a series of constant weights to terms:

$$P(\beta_i) = C_1 P(a, b) \cdot C_2 P(b, c) \cdot C_3 P(c, d), \qquad \beta_i = \{a, b, c, d\}$$

Whether weighting will be required of not is a topic of experimentation dependant on typical input characteristics; for this project, a prototype of the above function will be implemented for testing, Bigrams generated, and comparisons made between Trigrams explicitly generated from the training text and Trigrams generated by application of the above function to the existing

Bigrams. If sufficient performance and accuracy can be achieved using the above function, that function will be implemented as an Ngram generator for all token sequences greater than two elements in length.

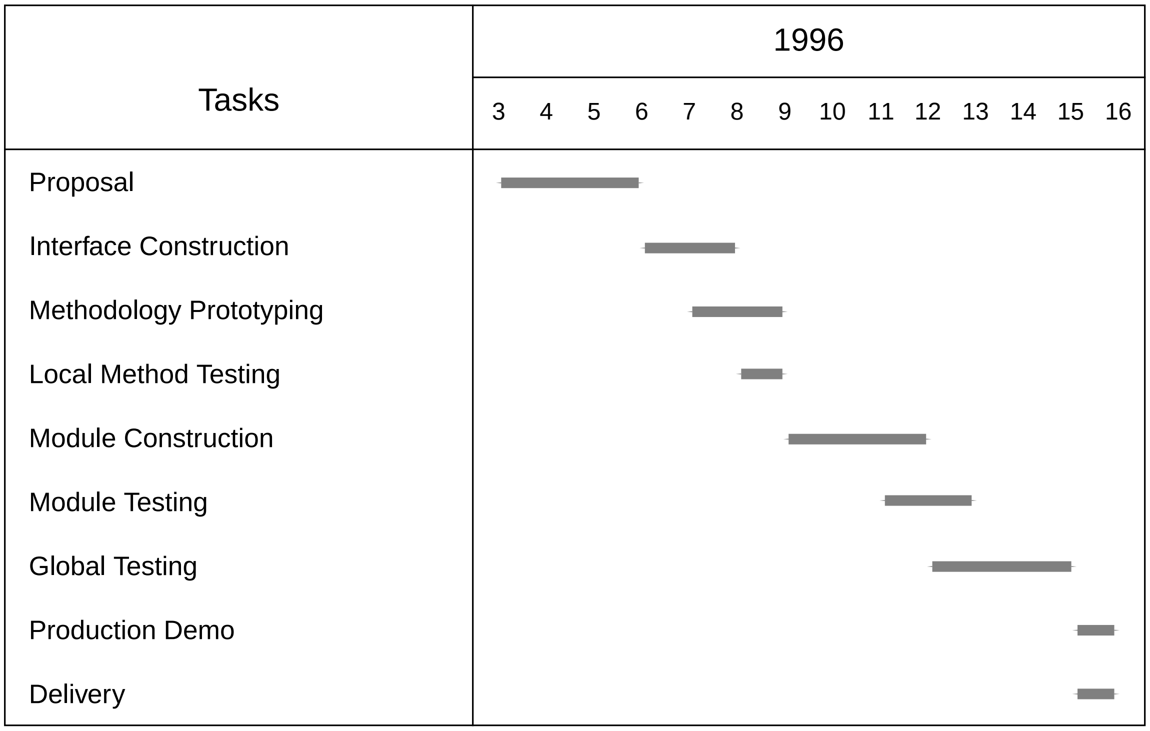
IV. EVALUATION

Unfortunately, little can be done to benchmark external performance of the LM independent of the CSR client; parallel implementations [if possible] of the CSR -- one with this project's LM and one with another LM -- can provide some means of performance measure relative to the two models, but the bulk of meaningful performance benchmarking must occur internally.

Input to the LM will be large body text, therefore the natural benchmark will be perplexity; all performance and results comparisons will be conducted with either identical texts or texts of near or equal resulting LM perplexity. The Ngram generation methodology will be tested in two stages; the first will verify output by comparing generated Ngram measures with calculated Ngram measures for identical Ngrams while the second will compare output of this project's LM with output of a LM produced independently.

Measures of performance will include relative distribution of fitnesses of LM hypothesis and first-order branching as a measure of perplexity; metrics of performance will include numeric measures of fitness for Ngrams and running time for specific model generation and query tasks. Change in coverage and perplexity for changes in volume of training text. though not necessarily useful in providing a measure of accuracy or error, will provide relative benchmarks for accuracy of the LM given varying degrees of granularity of the training text's representative domain.

V. SCHEDULE



VI. REFERENCES

1. (Jelinek et al., 1985)
2. (Deller et al., *Discrete-Time Processing of Speech Signals*, chapter on statistical language models, chapter on linear prediction)
3. (*Introducing ISIP* -- how to reference on-line materials?)
4. (*CMU Stat. LM Toolkit* -- how to reference on-line materials?)
5. (Lee and Alleva, *Continuous Speech Recognition*, pp. 634-635)
6. (Ronald Rosenfield, *Optimizing Lexical and N-gram Coverage...*)
7. (Picone, *Context-Sensitive Statistical Signal Processing: Toward...*, presentation)
8. (Picone, *Continuous Speech Recognition Using Hidden Markov Models*, IEEE ASSP July 1990)

proposal for

Development of an N-Gram Based Language Model for Continuous Speech

submitted to fulfill the semester project requirement for

EE 8993: Fundamentals of Speech Recognition

February 1, 1996

submitted to:

Dr. Joseph Picone

Department of Electrical and Computer Engineering
413 Simrall, Hardy Rd.
Mississippi State University
Box 9571
MS State, MS 39762

submitted by:

S. Given

Department of Electrical and Computer Engineering
Mississippi State University
Box 9571
Mississippi State, Mississippi 39762
Tel: 601-325-3149
Fax: 601-325-2298
email: given@ee.msstate.edu

I. ABSTRACT

An essential element of any speech recognition system is the language model. A language model attempts to identify and make use of the regularities in natural language to better define language syntax for easier recognition. One major obstacle in speech recognition is variability and uncertainty of message content. This, coupled with inherent noise, distortion and losses that occur in speech, emphasize the need for a good language model[1].

Several different types of language modelling techniques exist. This project will concern itself mainly with statistical language modelling. Statistical language modelling uses large amounts of text to automatically compute the model's parameters. This is called training. Language models can be compared using standard measures such as perplexity and recognition or word error rate. This project will use perplexity as a benchmark.

A good language model will provide *a priori* probabilities for all possible queries that the search algorithm may request pertaining to the learned vocabulary. Hence, the complexity of the model is directly related to the size of the corpus upon which it is trained.

II. INTRODUCTION

This project will attempt to build an accurate language model to be used in a continuous speech recognition (CSR) system. The proper interfaces between modules(front-end, language model, and search algorithm) must be implemented. The language model to use must be determined. This language model must properly model the training corpus and it must also utilize a method for handling outliers(words or phrases that occur infrequently, possibly never, in the training data). Some method for evaluation must be decided upon.

The speech recognition problem is of major importance in the development of more "human" like computers. It is desirable for the computer to understand and act upon vocal commands. This desire is derived from the fact that oral communication is the most common type of communication among humans and therefore most are highly skilled at communicating ideas through speech[5]. Since the majority of end users of computers are not fluent in any computer language the drive is to make the computer fluent in human language. If this push toward more "user-friendly" computing resources is successful, the applications are only a function of ones ability to imagine. Speech recognition has many areas of commercial applications such as: dictation, personal computers, automated telephone services, and special purpose industry applications[5].

Current dictation systems are designed to operate in office settings with head-mounted noise cancellation microphones. They are also speaker dependent as it is expected that one speaker will be using the system for an extended length of time. The two types of dictation systems are "unrestricted", which are used for letter writing or newspaper articles, and structured systems which are used for things such as report generation where the vocabulary is restricted to a certain type of report(medical, insurance, etc.). Current vocabulary sizes for dictation systems are about 40,000 words[5].

More people are exposed to computers now than ever and the trend toward silicon is ever increasing. To this end the goals of speech recognition are not only to make computers more easy for new users but also to make them more efficient for old pros. The ideas of changing font in mid-keystroke or opening a file by spoken command appeal to people who are on computers for the majority of every day[5]. Some people even suggest that with the trend toward smaller packages, the keyboard may be the limiting factor in size. This school of thought has visions of a totally speech driven interface.

Telephone-based recognition has potential in the areas of banking, credit card application and validation, shopping by catalog, and various customer service avenues. The main problem with this type of system is the uncertainty of conditions of use such as handset and microphone differences, channel noise, and low signal bandwidth[5].

Some other applications for speech recognition are industrial automation and providing necessary interfaces for people with disabilities. The main reason for success in industry is the marked increase in productivity in applications in which recognition systems help or replace human workers[5].

The table below shows the progression of speech recognition over approximately two decades[5]. Some helpful acronyms are: SI-speaker independent, SD-speaker dependent, CSR-continuous speech recognition, and IWR-independent word recognition.

Task	Late 70's	Mid 80's	Early 90's
SI IWR Alphabet	30%	10%	4%
SI CSR Digits	10%	6%	0.4%
SD CSR Query, 1,000 word (perplexity 6)	2%	0.1%	----
SI CSR Query, 1,000 word (perplexity 60)	----	60%	3%
SD IWR Dictation, 5,000 word	----	10%	2%
SI CSR Dictation, 5,000 word	----	----	5%
SI CSR Dictation, 20,000 word	----	----	13%

Table 1: Progress in speech recognition, as expressed by word error rate.

Other CSR systems have been designed and are constantly being improved, but this is an attempt to deliver a good, public domain, non-proprietary, CSR system that is accurate and easy to train. There is currently no such system available.

III. PROJECT SUMMARY

Natural language can be viewed as a stochastic process with each unit of speech(in our case words) being a random variable with some probability density distribution[1]. Given some speech signal, S, we would like to form some hypothesis as to what gave rise to that particular signal. In other words, given S, what word, L, corresponds to that particular signal. It is the job of the language model to provide a set of probabilities that effectively rank the hypothesis that it is given. That is, the language model will give the probability of some future word given a history of previous words that were spoken[3].

1. N-Gram Statistical Model

The n-gram model uses the previous (n-1) words as the only information source to generate the model parameters. N-grams are easy to implement, easy to interface with, and good predictors of short term dependencies, and thus have become the model of choice among statistical language models[1].

We can view n-grams from the approach that given any state (w_k,w_{k+1}), we will proceed to state (w_{k+1},w_{k+2}) with probability $P(w_{k+2}|w_{k+1}w_k...w_{k-(n-3)})$[2]. There is always a trade-off between reliability and detail. This is completely dependent on the size of the training data. The larger the training data, the more likely a large N will yield acceptable results. A smaller training corpus will necessitate a smaller choice in N in order to be reliably sure that an n-gram will exist. In cases where reliability becomes a problem some type of backoff approach can be used by the search algorithm.

Let's view this approach mathematically. The recognizer would like to find some word sequence, w^s, such that

$$w^s = \underset{w}{\arg max} P(w|y)$$

where w is any word string and y is the string of observations.. It is the job of the language model to provide the *a priori* information of the training corpus.

One of the goals of this language modelling project is to supply the search algorithm with an N-Gram statistical model as described above.

2. Deleted Interpolation

Regardless of the size of the training corpus, it is impossible to cover all possible words that the model will be queried for ranking. It is, however, desirable that the outliers that do not occur in the training data be accounted for in some manner. There are several methods for dealing with this apparent contradiction and avoiding the possibility of assigning zero probability to any words or word sequences. Setting some type of floor for ranking is one solution, another is deleted interpolation. These methods are also called smoothing.

Deleted interpolation is a method for supplementing training data that is insufficient for the speech recognition application for which it is being used[4]. This methods uses both tied and untied models properly weighted to lower the number of parameters and thus the variance of the models distributions. It has the effect of insuring that no zero probabilities are assigned. The affect of assigning zero probabilities will be discussed in the evaluation section. Another goal of this language model will be to use deleted interpolation to account for deficiencies in the training corpus.

IV. EVALUATION

3. Perplexity

Perplexity is a measure of performance of a speech recognition system. Care must be taken when discussing perplexity as it depends on the model and the training data. One can only get valid comparisons when all of these factors are taken into account. In particular the training data must be the same for the perplexity to have any real meaning at all.

Perplexity can be viewed as the geometric mean of the branch-out factor of the language[1]. When viewed from this definition the importance of comparing perplexities based on the same training corpus becomes obvious. A small training corpus would result in fewer branches at each node and hence lower perplexity than that of a larger corpus. The perplexity can be defined as[4]:

$$Q(\underline{w}) = 2^{\hat{H}(\underline{w})} \approx \frac{1}{\sqrt[N]{\hat{P}\left(w_1^N\right)}}$$

where H($\underline{w}$) is the entropy. It can be seen that if a zero probability were ever assigned in the model, an infinite perplexity would arise. This illustrates the need for some type of smoothing in order to be certain that outliers will be properly modelled. A low perplexity is not a sufficient condition to guarantee a low word error rate[3]. We will however use perplexity as the measure for success of this language model as it is uncommon for a low perplexity to lead to a high word error rate.

V. SCHEDULE

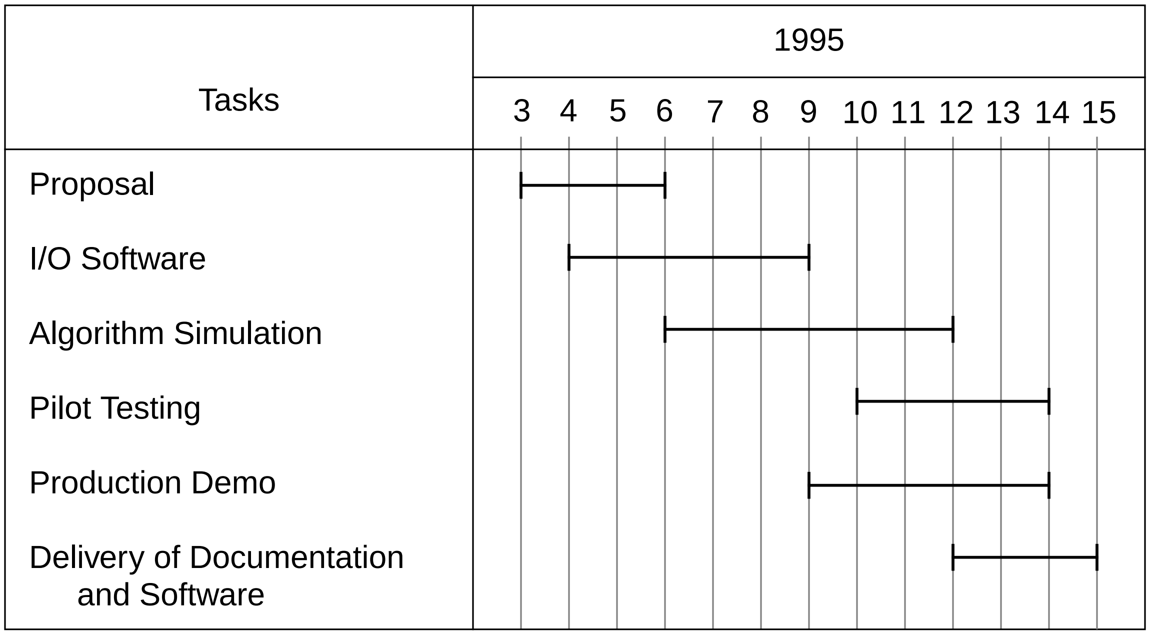


Figure 1. A time-line displaying the projected schedule for the language modelling section of the EE8993 continuous speech recognition system.

VI. REFERENCES

9. Rosenfeld, R. *Adaptive Statistical Language Modelling: A Maximum Entropy Approach*, Ph.D. thesis, Carnegie Mellon University, April 1994.
10. Lafferty, S. and Suhm, B., *Cluster Expansions and Iterative Scaling of Maximum Entropy Language Models*, in Fifteenth International Workshop on Maximum Entropy and Bayesian Methods, Kluwer Academic Publishers, 1995.
11. Koppleman, J., *A Statistical Approach to Language Modelling in the ATIS Domain*, MIT Department of Electrical Engineering and Computer Science, January, 1995.
12. Deller, John R., Proakis, John G., and Hansen, John H.L., *Discrete-Time Processing of Speech Signals*. Macmillan Publishing Co., New York, 1993.
13. Rudnicky, Alexander T. and Hauptmann, Alexander G. *Survey of Current Speech Technology*. School of Computer Science Carnegie Mellon University, Pittsburg PA, 1993.
14. http://sls-www.les.mit.edu/SLSpubs.html
15. http://www.cs.cmu.edu/
16. http://www.ri.cmu.edu/
17. http://www.mambo.ucsc.edu
18. http://www.speech.cs.cmu.edu/
19. http://www.ll.mit.edu