

Improvements in Audio Processing and Language Modeling in the CU Communicator

Jianping Zhang, Wayne Ward, Bryan Pellom, Xiuyang Yu, Kadri Hacioglu

Center for Spoken Language Research
University of Colorado, Boulder
Boulder, Colorado 80309-0594, USA

<http://cslr.colorado.edu>

Abstract

This paper presents some up-to-date audio processing techniques which have been developed and integrated into the University of Colorado (CU) communicator system. The CU Communicator is an interactive human-machine dialogue system for airline, hotel and rental car information. The baseline system was fully functional in June 1999. Since then, many improvements have been made. The paper will concentrate on acoustic echo cancellation, voice activity detection (VAD) and language modeling techniques and provide a paradigm for speech and audio processing in a dialog system with barge-in capabilities. Specifically, a real-time block least-mean-square (LMS) algorithm is discussed. A robust voice activity detector using energy threshold is applied to detect user voice. Experimental results are presented and some real-time implementation issues are addressed.

1. Introduction

In April 1999, the speech group at the University of Colorado implemented the CU communicator system. The baseline system was first demonstrated at the June 1999 DARPA Communicator meeting. Since then we have set up a dial-up Communicator system for data collection. To date, our system has about 350 registered users. 1750 calls totaling over 25000 utterances have been collected [1].

The CU Communicator system [10] consists of a hub and several servers that interact with each other through the hub. The user's voice from the telephone is first pre-processed by the audio server and passed to the speech recognition server through the hub. The pre-processing operation includes echo cancellation and speech activity detection. The speech recognition results are sent to the natural language processor to obtain a semantic parse of the recognition results. The dialogue management server generates SQL queries according to the interpretation given by the natural language processor and receives query results from the web via database server. Text-to-speech synthesizer will synthesize speech according to the query result text. Synthesized speech is output to the user via audio server [8].

During the months of June and July of 2000, the CU communicator together with systems from AT&T, IBM, BBN, SRI, CMU, MIT, Lucent and MITRE, participated a multi-site data collection effort conducted by the National Institute of Standards and Technology (NIST). Our system achieved a word er-

ror rate of 26.0% and task completion rate of 73.6%. An error analysis can be found in [1].

Careful examination of the NIST data revealed several deficiencies in the baseline system. First, the system enables recording after playing a tone to the user. If a user speaks before the tone, only partial utterances can be recorded and recognized. This causes increased recognition errors, especially on short utterances. To solve the problem, we needed to implement a Communicator with barge-in capabilities. Second, in the presence of loud background noise, the system may not respond quickly enough or even not respond at all after a user speaks. This is because the system can not detect the user's voice in loud noise. In such a case, a more robust voice detector is required. Third, the system has to improve its recognition accuracy and task completion rate to meet user requirements.

All these problems will be addressed in the following sections. Section 2 introduces the theory and implementation of barge-in audio processing. Specifically, a fast normalized least-mean-square (LMS) algorithm is realized to cancel echoes. Section 3 describes how we detect voice in the presence of echoes and noise. Improvements in the language modeling for recognition are given in Section 4. Brief conclusions are given in Section 5.

2. Audio pre-processing and echo cancellation

Echoes are caused by the imbalance between the two-wire subscriber loop and the four-wire hybrid in a telephone connection. In a barge-in audio server, the system asks questions and the user answers them according to the system prompts. A user may cut through during system speech. The recorded data is the superposition of the user's voice, echoes from system prompts and noise. The audio server must detect when the user speaks. If the user speaks when the system is playing prompts, then the playing needs to be stopped immediately. In order to detect the user voice, the system needs to start recording while playing is started. Simultaneously, the echo signal must be canceled from the recorded signal using the system speech as the reference signal. The resulting signal (error signal) is applied for voice activity detection. All these operations have to be performed block by block. The block size needs to be small enough to minimize the delay.

Broadly speaking, echo cancellation algorithms can be classified into two types: least-mean-square (LMS) and recursive-least-squares (RLS). The LMS algorithm is a type of stochastic gradient algorithm and the RLS algorithm can be viewed as a special case of the Kalman filter. The LMS algorithm variants

This work supported by DARPA through SPAWAR under grant # N66001-00-2-8906

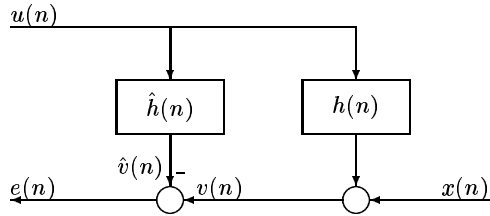


Figure 1: Echo canceler diagram

include Normalized LMS algorithm, Block LMS algorithm in which tap weights are updated once in each block, Proportionate Normalized LMS algorithm [2], and Fast LMS algorithm in which FFT techniques are used in computing echoes and new tap weights, etc. The RLS variants include QR-RLS algorithm which uses QR decomposition-based RLS algorithm and the QRD-LSL algorithm which uses QR decomposition-based least-square lattice filter [4]. Comparing the RLS with the LMS, the former typically has faster convergence rate and the convergence rate is invariant to the eigenvalue spread of the correlation matrix of the input speech. However, the simplicity and robustness of the LMS algorithm makes it attractive in practical systems.

2.1. Fast normalized LMS algorithm

In the CU Communicator, we use an adaptive filter to perform echo cancellation. Special requirements for the filter are real-time performance and fast convergence. A block diagram is shown in Fig. 1.

Let $u(n)$ denote system speech and $x(n)$ denote user voice (n is a time index). The $v(n)$ is the sum of $x(n)$ from the user and an echo from the channel with an unknown impulse response $h(n)$. If $\hat{h}(n)$ is a good estimate of $h(n)$, then $\hat{v}(n)$ approximates the echo portion of $v(n)$ and $e(n)$ is the error signal which is about equal to the speech $x(n)$.

A classic normalized LMS algorithm is performed in three steps:

1. Echo calculation

$$\hat{v}(n) = \sum_{k=0}^{M-1} \hat{h}_k(n) u(n-k) \quad (1)$$

where M is the number of taps.

2. Error estimation

$$e(n) = v(n) - \hat{v}(n) \quad (2)$$

3. Tap-weight adaptation

$$\hat{h}_k(n+1) = \hat{h}_k(n) + \mu \frac{e(n)u(n-k)}{\sum_{i=0}^{M-1} E[|u(n-i)|^2]} \quad (3)$$

where $k = 0, \dots, M-1$. The step-size parameter μ satisfies

$$0 < \mu < \frac{2}{\sum_{i=0}^{M-1} E[|u(n-i)|^2]} \quad (4)$$

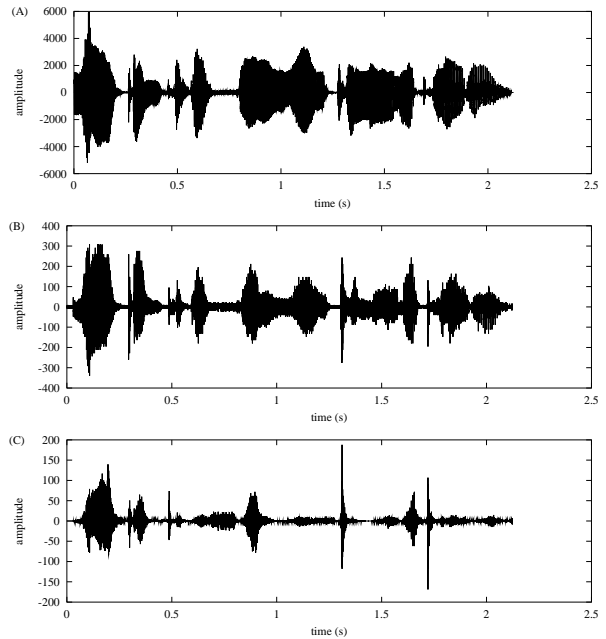


Figure 2: Waveforms of echo cancellation results. From top to bottom (A) reference system speech $u(n)$ of the utterance ‘‘Welcome to CU Communicator’’, (B) echo signal $v(n)$, (C) error signal $e(n)$, computed with $M = 256$, $L = 8$.

In our implementation, instead of updating tap weights at each time n , we update \hat{h} over a block interval of length L . So Equation (3) is changed to:

$$\hat{h}_k(n+1) = \hat{h}_k(n) + \beta \sum_{i=0}^{L-1} e(n-i)u(n-i-k) \quad (5)$$

where β is a normalized step-size parameter. The algorithm converges to the optimum Wiener solution as the number of iterations approaches infinity if β satisfies

$$0 < \beta < \frac{2}{\lambda_{max}} \quad (6)$$

where λ_{max} is the largest eigenvalue of the correlation matrix of the signal $u(n)$.

2.2. Some considerations

We have presented some equations to implement a block normalized LMS (NLMS) algorithm. However, for the Communicator system, we need to select appropriate values for the tap-weight number M and the update interval L . A typical choice is $M = 128, 256$ and $L = 8, 16$. Waveforms of system speech $u(n)$, echoed speech $v(n)$ and echo canceled speech $e(n)$ are shown in Figure 2. In the figure, the end user does not speak. So $x(n) = 0$ ($n = 0, 1, \dots$). The $\hat{v}(n)$ is an estimate of $v(n)$. When the system is playing prompts, the $v(n)$ is the sum of the echo signal and the noise signal in the phone line. In our practical implementation, we have addressed the following problems:

- Select an appropriate initial value for \hat{h} . If we can get a good estimate of the channel characteristics, then the algorithm can cancel echoes more efficiently. We have tried to play a cosine signal at a specified frequency when

the Communicator system just goes off hook and then record the echoes to obtain initial tap weights. However, such obtained tap weights are not very useful and the algorithm will take almost the same time to reach the convergent point as with all zero initial value. Our experiments show that playing white noise is a good way to obtain initial tap weights. The problem is that the noise will make end user uncomfortable when the user dials into our system. In our system, a zero initial value was used for \hat{h} .

- Decide the echo delay. We found if we could calculate the channel delay in advance, then the delay information would help the echo canceler. In a real system, playing and recording operations can not be started simultaneously. So there will also exist a hardware delay. It is possible to measure the delay using a cross-correlation method.
- Attenuate the discontinuity. The discontinuity in the waveform means there are high frequency components in the power spectrum. In such case, the echo canceler does not work very well. We can see that near the sampling time 1.32s and 1.71s, the echo signal has not been efficiently canceled. This requires the application of a window to attenuate the discontinuity.

3. Voice activity detection

Deciding whether an end user is speaking is the second challenging aspect in the Communicator. Efficient voice activity detection (VAD) algorithms need to be developed to perform starting point detection. The algorithm must avoid false triggering. Currently, approaches using energy threshold, zero crossing rate and least-square periodicity estimator have been proposed [3]. The Communicator uses the classic energy threshold method. In this method, the speech signal is blocked into frames of N samples. Usually, for speech digitized at a frequency of f (Hz), $0.025 \leq N/f \leq 0.05$ is chosen. Unlike the LPC or MFCC feature extractor of a speech recognizer, there is no sample overlap between adjacent frames. The l^{th} frame energy is calculated as

$$E(l) = \sum_{i=0}^{N-1} |e(Nl + i)|^2 \quad (7)$$

where $l \geq 0$. The term $e(Nl + i)$ can be obtained from (2).

We need to pay attention to energy computation and threshold estimation:

1. The error signal is used to compute frame energy. Due to the convergence rate of the echo canceler, the beginning $0.5 \sim 1s$ samples need to be skipped. We have also tried a short-term power method by comparing the estimated power of $v(n)$ with $\hat{v}(n)$. If the estimated power of $v(n)$ is greater than the largest power of $\hat{v}(n)$ within the past N samples, then a speech event is triggered. Comparing the short-term power method with the above energy threshold method, the former is fast but the latter is robust.
2. Carefully select threshold. If the energy threshold is high, then a user will feel difficult to cut through. Otherwise, the false triggering rate will be high. In the Communicator, the energy threshold is adapted dynamically with the current energy threshold and all the past ones so that impulse noise can be smoothed.

In the next Section, we consider improvements in language modeling in the CU Communicator.

4. Improvements in language modeling

First we will introduce some language modeling techniques.

4.1. Review of language modeling techniques

The fundamental goal of a speech recognizer is to find the best word sequence \hat{W} based on the acoustic observation sequence Y . It can be expressed as [7]:

$$\hat{W} = \arg \max_W P(W|Y) \quad (8)$$

Using Bayes' Rule, the term $P(W|Y)$ can be written as:

$$P(W|Y) = \frac{P(Y|W)P(W)}{P(Y)} \quad (9)$$

Since $P(Y)$ is independent of W , Equation (8) becomes

$$\hat{W} = \arg \max_W P(Y|W)P(W) \quad (10)$$

The first term $P(Y|W)$ in (10) is called the acoustic model and the second term $P(W)$ is called the language model.

The Communicator uses the CMU SPHINX-II recognizer. Like many other ones, the recognizer has to make a trade-off between speed and recognition accuracy. Two schemes exist in modern recognizers, one-pass schemes and two- or multi-pass schemes. In one-pass schemes, powerful language models are used in the single decoding pass. As a result, the optimal word sequence can be obtained in a frame synchronous way. However, Since the language model is applied for every possible word combination in the search path, the algorithm is usually much slower than that of two- or multi-pass schemes. In two- or multi-pass schemes, a simplified language model is used in the first pass. The aim of the first pass is to decode the speech sequence into a word lattice. More powerful language models are applied in the second pass to search the word lattice and find the optimal word sequence. Compared to the one-pass scheme, the latter is more efficient.

Since a language model is indispensable in a recognition system, various language modeling techniques have been proposed. Among them, word-based n -gram language models are most widely used. Because it is difficult to capture long-distance word constraints with an n -gram, other models such as maximum entropy [5] and latent semantic analysis [6] have been developed. Since n -gram probabilities are difficult to estimate reliably for $n \geq 2$ when trained from very sparse data, class-based language models are more realistic in this case [7]. Also, to port a language model from one domain to another domain, language model adaptation is often applied. In our Communicator, class-based language models are used.

In language modeling, perplexity is calculated to evaluate the performance of a model given a corpus. The definition of language model perplexity pp is given below:

$$pp = P(w_1, w_2, \dots, w_Q)^{-1/Q} \quad (11)$$

where Q is the total number of words in the corpus. The $P(w_1, w_2, \dots, w_Q)$ is the probability of the word sequence w_1, w_2, \dots, w_Q , given the language model.

Original utterance:

I want to go from BOSTON to PORTLAND at NINE A.M.

Tagged utterance:

I want to go from [city:BOSTON] to [city:PORTLAND]
at [hour_number:NINE] [am_pm:A.M].

Figure 3: A tagged utterance

4.2. Language modeling improvement

The Communicator system is designed for end users to get up-to-date world-wide air travel, hotel and rental car information via the telephone. There are many names for countries, states, provinces, cities, airlines. To train a robust language model, names are clustered into different classes. An utterance with class tagging is shown in Fig. 3. In the figure, city, hour_name and am_pm are class names.

The probability of word w_i given class c_i is estimated from training corpora. After the corpora are correctly tagged, a class-based trigram language model can be computed from the tagged corpora as follows:

- Estimate class transition probabilities $P(c_i|c_{i-2}, c_{i-1})$. The class transition probabilities can be obtained as:

$$P(c_i|c_{i-2}, c_{i-1}) \cong \frac{F(c_{i-2}, c_{i-1}, c_i)}{F(c_{i-2}, c_{i-1})} \quad (i > 2) \quad (12)$$

Where $F(c_{i-2}, c_{i-1}, c_i)$ is the count of the number of times that the word sequence (c_{i-2}, c_{i-1}, c_i) appears in the tagged corpora.

- Estimate observation probability $P(w_i|c_i)$. The probability of word w_i given class c_i is:

$$P(w_i|c_i) \cong \frac{F(w_i, c_i)}{F(c_i)} \quad (i > 0) \quad (13)$$

Note that $F(c_i)$ is the frequency of the class c_i in the tagged corpora.

Given the above equations, the equation for calculating perplexity of class-based trigram language models can be derived from Equation (11). After $P(w_i|c_i)$ is trained from our tagged corpora, we use our testing data. Perplexity is decreased from 23.84 to 20.19. The perplexity of 23.84 is calculated by assuming uniform distribution among all the words in a class. As a result, recognition accuracy using one-pass schemes has increased from 84.2% to 84.6%.

To improve the robustness of our class-based language model, different discounting methods have been tested to address the zero-frequency problem. We know that even if the training corpora are large, zero-frequency items may occur. It is not easy to gather enough training data to train a robust language model. So it is necessary to try various discounting methods and select an appropriate one for our system. Four methods are evaluated. They are: Witten-Bell, Good-Turing, linear and absolute. The theory aspects of these discounting methods were investigated in [9].

The experimental results are summarized in Table 1. In the experiment, 2191 sentences uttered by 10 male speakers are used for testing data. The overall perplexity of the testing corpus is 18.95. The first column is male speaker number. Number of sentences uttered by the corresponding speaker is shown in the second column. Word error rates (in percentage) for four

discounting methods are given in columns 3 to 6. The last two rows are average word error rate (averaged over sentences) and mean word error rate (averaged over speakers).

Table 1: Results of different discounting methods

spkr	#snt	Witten-Bell	Good-Turing	linear	absolute
m0	34	12.4	15.7	12.4	14.4
m1	18	24.2	27.3	27.3	27.3
m2	41	19.9	22.9	23.3	22.9
m3	26	20.4	20.4	19.8	19.8
m4	9	21.2	21.2	21.2	21.2
m5	46	11.8	11.1	14.1	11.8
m6	1470	9.7	9.4	10.5	9.4
m7	193	6.3	6.3	6.0	6.3
m8	50	8.2	8.7	10.0	10.0
m9	304	11.4	11.6	11.6	10.9
sum/avg	2191	10.2	10.2	11.0	10.1
mean	219.1	14.5	15.5	15.6	15.4

From Table 1, we conclude that in the mean sense, Witten-Bell method is the best one.

5. Conclusions

We have improved the CU Communicator system by adding barge-in capability and fine tuning the class language model. Although the new audio server operates in the linux environment and uses Dialogic D/4IESC, it is easy to port to other unix operating systems and Dialogic cards.

6. References

- [1] Pellom, B., Ward, W., and Pradhan, S., The CU Communicator: An Architecture for Dialogue Systems, ICSLP, Beijing, 2000.
- [2] Duttweiler, D., "Proportionate Normalized Least-Mean-Squares Adaptation in Echo Cancelers", IEEE Trans. Speech and Audio Proc., 8(5):508-517, 2000.
- [3] Tanyer, S., and Özer, H., "Voice Activity Detection in Nonstationary Noise", IEEE Trans. Speech and Audio Proc., 8(4):478-482, 2000.
- [4] Haykin, S., Adaptive Filter Theory, Prentice-Hall, Inc., New Jersey, 1996.
- [5] Rosenfeld, R., "A Maximum Entropy Approach to Adaptive Statistical Language Modeling", Computer Speech and Language, 10:187-228, 1996.
- [6] Bellegarda, J., "A Multispan Language Modeling Framework for Large Vocabulary Speech Recognition", IEEE Trans. Speech and Audio Proc., 6(5):456-467, 1998.
- [7] Rabiner, L. and Juang B. H., Fundamentals of Speech Recognition, Prentice-Hall, Inc., 1993.
- [8] Ward, W. and Pellom, B., The CU Communicator System, Automatic Speech Recognition and Understanding Workshop, Colorado, 1999.
- [9] Chen, S. and Rosenfeld, R., "A Survey of Smoothing Techniques for ME Models", IEEE Trans. Speech and Audio Proc., 8(1):37-50, 2000.
- [10] <http://communicator.colorado.edu>