A ROBUST SPEAKER–INDEPENDENT CPU–BASED ASR SYSTEM

R. Obradovic, D. Pekar, S. Krco, V. Delic, V. Šenk
School of Engineering, University of Novi Sad, Yugoslavia
tlk_delic@uns.ns.ac.yu

Abstract

In this paper a new automatic speech recognition (ASR) CPU-based software, called AlfaNum, with the chosen few heuristics optimized for applications in heterogeneous conditions is described. AlfaNum is a discrete speaker-independent ASR product intended for application in the largest bank-by-phone interactive voice response (IVR) system in Yugoslavia, with a lot of customers all over Serbia. That means a large variety of dialects, telephone line quality, and microphones used.
This system has been tested on 500 speakers and it achieved an average accuracy of 98,2% in real life conditions. The whole software is developed in C++ programming language. Object oriented programming gave the software an elegant look, and minimized all possible errors. On the other hand, the power of C++ language and its tight interaction with machine made the software fast and efficient.

1. INTRODUCTION

AlfaNum is a HMM-based speech recognizer, designed as software-only solution that requires no additional dedicated ASR hardware. It runs on Pentium-class processors, supports the Windows NT operating system and can operate in conjunction with several Dialogic voice boards, providing a cost-effective way to add ASR capability to CTI applications.
It is intended for application in the largest bank-by-phone interactive voice response (IVR) system in Yugoslavia, with a lot of customers all over Serbia. That means a large variety of dialects, telephone line quality, and microphones used. Moreover, the set of ten digits, 0 – 9, is a difficult vocabulary in Serbian because of conflict digit pairs (e.g. 1=JEDAN, 7=SEDAM), and plenty of dialects (e.g. 4=CETIRI=CETRI=CETIR).
A lot of comprehensive experiments including their visualization are done. Along with the standard ASR methods in isolated word recognition some original features, which will be described later, were also used.
As in any ASR system there are standard tasks that should be performed in order to accomplish successful word recognition:

- determining word boundaries,
- feature extraction,
- effective training,
- right model adoption,
- fast and effective recognition.

In the next sections we will describe these steps.

2. DETERMING WORD BOUNDARIES

The basis for the word bounding is the short-term signal power. Signal is divided into frames, each 30ms long and 15ms shifted from the previous one.
The average power in each frame is calculated as

$$P(n) = \frac{1}{N}\sum_{i=0}^{N-1} s^2(i + N \cdot n). \qquad (1)$$

Signal and noise power are determined after a percentage of frames with the smallest and the highest power are rejected. The estimated noise power is then equal to average power of the weakest remaining frame. The signal power equals the average power of the strongest remaining frame reduced by noise power. The estimated signal to noise ratio is

$$SNR = 10 \log \frac{P_S}{P_N}. \qquad (2)$$

If SNR is less than some predefined value (10dB), that signal is declared to be unusable for recognition.
Neighboring frames are grouped into clusters which differ no more than few decibels (e.g. 5dB). The signal is thus divided into few clusters with very small variations of average power within cluster. Each combination of neighboring clusters is treated as a possible word and is called the hypothesis. The number of such hypothesis is

$$N_H = \frac{c \cdot (c+1)}{2}, \qquad (3)$$

where c is number of clusters.
Scoring is performed for each hypothesis. There are few heuristics which are used for scoring. Each of them has its own coefficient, experimentally determined:

- energy of all frames in that hypothesis $\left(score+ = K_E \cdot \sum_i E(i)\right)$.
  Logically, the strongest part of the signal is the most likely to be the word.
- overstep of some predefined maximum length $\left(score- = K_{d1} \cdot (l - l_{\max})^2, l > l_{\max}\right)$.
  This heuristics is introduced primarily in order to prevent algorithm to favorize longer hypothesis.
- understep below some predefined minimum length $\left(score- = K_{d2} \cdot (l - l_{\min})^2, l < l_{\min}\right)$.
- silence before, after and between vowels, where the strongest clusters are considered as vowels $\left(score- = K_{v(1,2,3)} \cdot \frac{l^2}{E_s}\right)$,
  where l is silence length, E_s the average silence

energy, and K is a ponder which is different for all three cases. The point is that the word always includes at least one vowel, and that there cannot occur a long period of time without it. Hypotheses that do not observe this rule receive negative scoring.

- signal energy at the very boundary of the word. $\left(score-= K_{b(1,2)} \cdot E^2\right)$

  where E is the signal energy at the beginning or ending of the word, and K_b is the appropriate coefficient. The idea is that the word boundary cannot be at the spot where the signal has significant energy. This is especially useful if the task is to extract few words from the sequence of connected words.

This method proved to be very efficient and robust, even in noisy environment. But, there is one drawback. When the word starts or ends with a very silent phoneme (e.g. "s"), it will probably be cut as non-word part of the signal. If we adjust coefficients K_{v1} and K_{v2} the silent phoneme could be included in the word, but this will inevitably induce that in some other cases unwanted noise is treated as a part of the word. And that phoneme is often crucial for recognition (sedam - jedan). Winning hypothesis is then expanded at the beginning and at the end for some constant amount of time, and those are estimated word boundaries. This makes process of estimating word boundaries semi-implicit, because some non-speech parts of the signal are included in recognition.

3. FEATURE EXTRACTION

A feature vector is calculated for each frame in the signal. Vector of size 36 is adopted consisting of:

- mel-cepstrum vector of size 14,
- frame energy,
- zero crossing rate (ZCR),
- degree of voicing,
- pitch.

Every feature has its delta value which covers the remaining 18 rows of the feature vector. More information about each of these features is given below.

3.1. Mel-cepstrum

Windowed signal is expanded to the smallest power of 2. After performing FFT, mel coefficients are calculated by

$$c(n) = \frac{2}{N}\sum_{k=1}^{N-1} S(I(k)) \cos\frac{2p}{N} kn, \qquad (4)$$

where N is the number of samples, and $I(k)$ is a conveniently chosen function which transforms the mel sample k into an adequate sample in the oversampled spectrum. $S(m)$ denotes the m-th sample of spectrum modulus logarithm.

3.2. Frame Energy

Frame energy is calculated by

$$E = \frac{\left(\frac{1}{N}\sum_{k=0}^{N-1} s^2(k) - E_{\min}\right)}{E_{\max}}, \qquad (5)$$

where $s(k)$ is the signal modulus, and E_{min} and E_{max} are minimal and maximal energy of the whole word, respectively. This ensures independence for this feature of signal intensity.

3.3. Zero crossing rate

ZCR represents the normalized count of signal sign changes per frame. If we define a (clipped) binary time series $X_1, X_2, ..., X_N$ by the nonlinear transformation,

$$X_t = \begin{cases} 1, & \text{if } s(t) \geq 0 \\ 0, & \text{if } s(t) < 0 \end{cases}. \qquad (6)$$

The number of zero-crossings, denoted by ZCR, is defined in terms of $\{X_t\}$,

$$ZCR = \frac{\sum_{t=2}^{N} [X_t - X_{t-1}]^2}{N-1}. \qquad (7)$$

3.4. Voicing and pitch

The typical speech signal and spectrogram is shown in Figure 1. It is easy to notice the characteristic structure of the spectrogram in the areas of vowels. Distance between two lines is the basic voice frequency, the so called pitch, which can vary from 50-400Hz. The presence of such a structure in signal is called voicing, and it clearly shows the position of vowels in signal. It would be useful to find a quantitative way to determine the degree of presence of such a structure in the signal. FFT of the spectrum modulus for each frame should be found. Samples between 50 and 400Hz in that new signal should be observed, and if the voicing structure is present, one of the samples will dominate. So the sample with the greatest value of the normalized, above mentioned signal is voicing and its position is pitch. The same value would be obtained if the linear scale cepstrum up to a high order were found and the greatest cepstrum component declared as voicing.

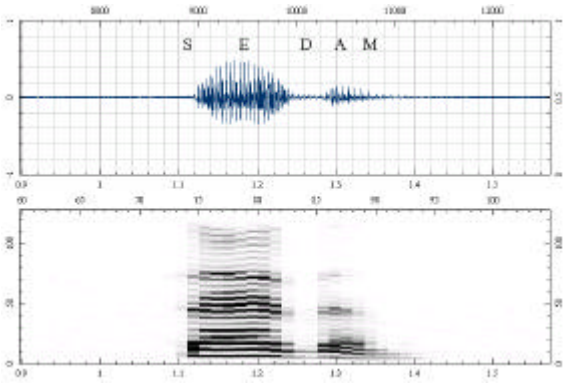

Figure 1 Signal and spectrogram of the word "sedam"

3.5. Delta values of features

Delta coefficients of all the features are calculated by

$$\Delta f_i(n) = f_{i+1}(n) - f_i(n), \quad (8)$$

These, practically new features, are very useful in recognition, since they evaluate the basic feature changes in time. But it is noticed that, due to oft and sudden changes in feature values, these delta values may vary a lot, even in the cases where the basic feature structure is very similar. There are a few ways of decreasing this negative influence. The resulting delta feature vector is filtered in time. Simple FIR filter is used,

$$\Delta f_i'(n) = a \cdot \Delta f_i(n) + b \cdot \Delta f_i(n+1) + c \cdot \Delta f_i(n+2). \quad (9)$$

This improved the recognition performance by almost 2%.

4. MODEL TRAINING

4.1. Maximum likelihood criteria

The maximum likelihood training criteria can be given by

$$\Lambda_{ML} = \arg\max_{\Lambda} P(Y \mid \Lambda_C) \quad (10)$$

where Y is the set of all observation sequences, Λ the set of parameters for all models, Λ_C the set of model parameters which concern one observation sequence. Λ_{ML} denotes the set of parameters for all models obtained by ML training. ML training was practically implemented by the Baum-Welch algorithm. Based on (10) after taking the logarithm, we find the expressions for calculating all parameters for each model,

$$\Lambda^i_{ML} = \arg\max_{\Lambda^i} \sum_{m=1}^{M_i} \log P(Y^i_m \mid \Lambda^i), \quad i = 1...N \quad (11)$$

where N is the number of models, Λ^i denotes the parameter set for model i, Y^i_m is the m-th sequence of the i-th model, M_i is number of sequences which correspond to model i, and Λ^i_{ML} are the parameters of the i-th model obtained by ML training. We see that parameters for each model are calculated exclusively from the observation sequences of that model, not taking other observation sequences into account. This means that ML training and decision making cannot be very good to make distinction between similar words.

4.2. Maximum mutual information criteria

In the previous paragraph we saw drawbacks of the ML training. The basic idea of the maximum mutual information (MMI) criteria is to overcome those drawbacks. After training one model and calculating its parameters, all available observation sequences should be taken into account, not only the ones of that word. In this approach, the correct model Λ_C is trained positively while all the other models are trained negatively on the observation sequence Y, helping to separate the models and improve their discriminating ability during testing. Mutual information between an observation sequence Y and the correct model Λ_C is defined as

$$I_\Lambda(Y, \Lambda_C) = \log P(Y \mid \Lambda_C) - \sum_i \log(P(Y \mid \Lambda_i) P(\Lambda_i)), \quad (12)$$

where the first term represents positive training on the correct model Λ_C (just as for ML), while the second term represents negative training on all the other models Λ_i. Training with the MMI criterion then involves solving for the model parameters Λ that maximize the mutual information,

$$\Lambda_{MMI} = \arg\max_{\Lambda} I_\Lambda(Y, \Lambda_C). \quad (13)$$

4.3. Modified MMI criteria

In (12) we see that the goal function increases both by increasing $P(Y|\Lambda_C)$ (the correct model) and decreasing $P(Y|\Lambda_i)$ (incorrect models). This is a reasonable criterion but it can be improved. If, for example, some models already have $P(Y|\Lambda_i)$ small enough and cannot be mixed with the correct model during recognition, it is not necessary to further decrease it. It is more important to concentrate on the models which did not yet achieve sufficient distinction (still have significant $P(Y|\Lambda_i)$). Therefore, we tried the modified goal function,

$$g(\Lambda) = -\sum_{i=0}^{N} \sum_{j=0}^{N} \sum_{m=0}^{M_i} e^{-s \cdot [\log P(Y^i_m|\Lambda_i) - \log P(Y^i_m|\Lambda_j)]}. \quad (14)$$

An intuitive explanation of such a goal function is as follows. In the brackets there is the difference between the probability of correct and incorrect model for one single incorrect model, and one single observation sequence. The total sum among all words, (incorrect) models and observation sequences gives the value of the goal function. If in one of the members of the sum, the difference in the brackets is rather large, there is no conflict between the correct and that particular incorrect model for that observation sequence. The influence of this member of sum onto the goal function should be minimal. That is exactly what the rest of the expression does. If the value in the brackets is large enough, then the value of that member of the sum is close to zero (s>0). If the difference is small or even negative, then the influence is huge, and training will concentrate on those cases. Parameter s represents "stiffness" of the training. It quantitatively represents how much more concern will be dedicated to critical cases. If it approaches 0, this becomes MMI training. It should be rather large, but the price is an increased dependency on the training set.

4.4. Implementation of corrective training

As with the MMI, model training involves solving for the model parameters Λ that maximize:

$$\Lambda^i_{MMMI} = \arg\max_{\Lambda} g(\Lambda), \quad i = 1...N. \quad (15)$$

The evolution strategies algorithm ES(1-1) [5] is used for training the models. since the problem is too

complex, suboptimal solution is found. First, initial models are obtained using ML training. Next, segmentations for all observation state sequences are found using the Viterbi algorithm. During one set of ES(1-1) iterations false assumption is that the segmentations will never change. It is a fairly acceptable presumption. After one set of iterations, new segmentations are found and the process continues.

5. MODEL STRUCTURE

System AlfaNum uses the modified version of Mixture Density Hidden Markov Models (MDHMM). Model structure is linear which means that in transition matrix a only members $a_{i,i}$ and $a_{i,i+1}$ are different than zero.

5.1. Mixture Density Hidden Markov Models

Conventional implementations of MDHMM-s recognition are based on the calculation of $P(O|\Lambda)$ or $P(O,Q|\Lambda)$, where denotes Q sequence obtained by the Viterbi algorithm. These values are calculated by

$$P(O \mid \Lambda) = \sum_{Q} P(O, Q \mid \Lambda), \tag{16}$$

$$P(O,Q \mid \Lambda) = p_{q_1} b_{q_1}(O_1) a_{q_1 q_2} b_{q_2}(O_2) .. a_{q_{T-1} q_T} b_{q_T}(O_T), \tag{17}$$

$$b_{q_t}(O_t) = \sum_{k=1}^{M} c_{jk} N(O_t - m_{jk}, U_{jk}), \tag{18}$$

after taking the logarithm we get

$$\begin{aligned} &\log f(O,Q \mid \Lambda) = \log p_{q_1} \\ &+ \log b_{q_1}(O_1) + \log a_{q_1 q_2} \\ &+ \log b_{q_2}(O_2) + ... + \log a_{q_{T-1} q_T} + \log b_{q_T}(O_T) \end{aligned} \tag{19}$$

Let's denote

$$f(O,Q,\Lambda) = \log P(O,Q \mid \Lambda). \tag{20}$$

The recognition is performed on the basis of

$$\max_{Q} f(O,Q,\Lambda), \tag{21}$$

where the sequence Q which maximizes the expression is obtained by the Viterbi algorithm.

5.2. State ponders

Another original contribution of this work is the so called "state ponders" involvement in the model structure. Sometimes we recognize one word from another solely relying on one part of the word, discarding the rest (because it is the same in both words). For example, in Serbian language words "sedam" and "jedan" have three identical phonemes, and one very similar (the last one). So, the recognizer should concentrate on the beginning of the words to successfully distinguish one word from another. It is impossible to do so, using the standard model structure for HMM. To overcome this drawback of the standard HMM, we added one more parameter per each state to the HMM, and called it state ponder, so that (19) becomes

$$\begin{aligned} &\log f(O,Q \mid \Lambda) = \log p_{q_1} \\ &+ s_{q_1} \log b_{q_1}(O_1) + \log a_{q_1 q_2} \\ &+ s_{q_2} \log b_{q_2}(O_2) + ... + \log a_{q_{T-1} q_T} + s_{q_T} \log b_{q_T}(O_T) \end{aligned} \tag{22}$$

where s_i is the ponder of the i-th state.

Besides the original idea and purpose for including state ponders in the model structure, there is one more merit of such a concept. It is well known that vowels last far longer than consonants and hence have greater influence on the above expression i.e. recognition metric. So, without state ponders, the training algorithm would concentrate on matching vowels thus neglecting consonants which are far more important in recognition. State ponders are estimated during corrective training with all other model parameters.

6. CONCLUSION

An isolated word, speaker independent ASR system is designed and developed, and is currently in test phase in real life conditions. Currently system performes with 98.2% digit recognition percentage. System includes some original features like new training criterion called Modified Maximum Mutual Information, and efficient training procedure based on evolution strategies algorithm.

REFERENCES

[1] Krco S. *Apendage to the Methods of Isolated Words Recognition,* **ETF,** University of Belgrade, 1997.

[2] Rabiner, B.H. Juang, *Fundamentals of Speech Recognition* , Prentice-Hall, New Jersey 1993.

[3] Tebelskis J. *Speech Recognition Using Neural Networks*, Ph.D. work, School of Computer Science, Carnegie Mellon University, Pittsburgh, 1995.

[4] Picone J. *Signal Modeling Techniques in Speech Recognition, 1992.*

[5] Back T., Schwefel H. P. *An Overview of Evolutionary Algoritms for Parameter Optimization, University of Dortmund,* 1993.