CONNECTED WORDS RECOGNITION

Darko Pekar, Radovan Obradoviæ, Vlado Deliæ
School of Engineering, University of Novi Sad, Yugoslavia

I INTRODUCTION
In [1] the same group of authors worked on the system for discrete digits recognition in Serbian language. Problems in that work are located, and many of them overcome and solved by methods proposed in this paper. New system not only surpasses previous in discrete word recognition, but it also gives very good results when applied to fast pronounced connected digits in adverse conditions (background noise, telephone channel etc.). New system is also speaker independent, works in real time, even on weak Pentium I configurations and supports Windows NT/9x operating systems. It does not require any additional hardware, can work with several Dialogic sound cards, which offers inexpensive way of providing CTI applications with ASR support.

II ACOUSTIC FEATURES VECTOR
Acoustic features vector consists of static and dynamic components.

Static features
In use are: 14 MFCC, signal energy, zero crossing rate and degree of voicing.

Dynamic features
In order to characterize feature change in time first derivative of the static features is used. We carried out experiments with two methods for first derivative calculation:
- difference between two neighbouring frames, filtered in time with a filter of length 3, which significantly reduced its noisy component [1],
- first derivative calculation using the regression coefficients.

Regression coefficients are obtained when the function is approximated with the polynomial using the Minimal Square Error (MSE) criteria, and then first derivative of that function is found at the desired spot. This is done in a window of certain length. If we use the second order polynomial we obtain the following expression:[6]

\[ r_{i,j} = \frac{\sum_{q=0}^{2Q+1} q P_{r,q,j}}{\sum_{q=0}^{2Q+1} q^2} \]  

(2.1)

where \( 2Q+1 \) stands for window duration (in frames), \( P_{r,q,j} \) is static feature value, \( i \) frame number, and \( j \) static feature index.

III MODEL STRUCTURE
Words are modeled with HMM which has simple linear structure, and each phoneme in word is modeled with two states. Acoustic vectors probability distribution during one state is modeled by a mixture of 6 Gaussian distributions. It is common to combine Gaussian distributions in the following way:

\[ p(x) = \sum_{k=1}^{K} c_k e^{-\frac{1}{2} (x - \mu_k)^T U_k (x - \mu_k)^T} \] 

(3.1)

where \( x \) represents the acoustic feature vector, \( \mu_k, U_k \) are the parameters of the Gaussian distributions, and \( k \) is the number of mixtures.

In our opinion this approach is insufficiently represented in ASR practice, although according to our results, can be much more efficiently implemented, because the software realizations usually use the logarithm of probability density function:

\[ \ln p(x) = \max_{k=1}^{K} c_k e^{-\frac{1}{2} (x - \mu_k)^T U_k (x - \mu_k)^T} \] 

(3.2)

In our opinion this approach is insufficiently represented in ASR practice, although according to our results, gives results comparable to first method, and can be much more efficiently implemented, because the software realizations usually use the logarithm of probability density function:

\[ \ln p(x) = \max_{k=1}^{K} c_k e^{-\frac{1}{2} (x - \mu_k)^T U_k (x - \mu_k)^T} \] 

(3.3)

We see that for expression (3.3) evaluation it is unnecessary to use transcendental functions, but only simple operations such as addition, subtraction and multiplication, which can be implemented very efficiently on DSP processors. We emphasize that the expression (3.3) evaluation is very similar to the algorithm of vector quantization.

IV MODEL PARAMETERS CALCULATION
All previous systems for discrete word recognition had more or less standard training procedure: initial model determination, Baum-Welch reestimation method by the Maximum Likelihood criteria (ML) [2], and some methods which perform discriminative training.

In the first step the initial distribution of the feature vectors among the states is carried out, in such a way that a similar accumulated difference between neighbouring frames within each state is preserved [2]. This method didn’t guarantee that one state would cover the same part of the same phoneme in every observation sequence. On the contrary, it was inevitable to have features from two or even more phonemes within one state. None of the later training methods couldn’t correct this starting error. This was especially expressed in the states in the middle of words, and had catastrophic consequences to recognition performance, because such states could "recognize" virtually anything. That is why our system requires segmented speech database. This implies that the information about each phoneme start and end point is provided for every observation sequence in the training set.
Manuel segmentation of sufficiently large speech database requires huge amount of work, so we developed some tools and techniques in order to automate it and speed it up. First, it is necessary to manually determine segmentation of some small portion of database. Initial model parameters are calculated out of that small database, and used for automatic segmentation of the rest of the database. At the end it is necessary to manually inspect and correct segmentation of suspicious samples. Based on this labels system builds models with 2 states per phone, and in the whole process of its training it is precisely determined which part of feature vectors (sequences) belongs to which state.

**Parameter estimation according to ML.**

Let us denote with \( X \) a set of all acoustic feature vectors which belong to the state with the set of parameters \( \Lambda \) according to segmentation. Suppose that all feature vectors have the same distribution and that they are statistically independent. Then we estimate model parameters by maximizing the likelihood function.

\[
\hat{\Lambda} = \arg \max_{\Lambda} p(X \mid \Lambda)
\]

Due to the same distribution and independence

\[
\hat{\Lambda} = \arg \max_{\Lambda} \prod_{i=1}^{N} p(x_i \mid \Lambda)
\]

(4.1)

where \( N \) stands for the number of words in the dictionary. If we assume that all the weight coefficients are equal and approximate matrix \( U \) with appropriately chosen known matrix \( L \) we get:

\[
\hat{\Lambda} = \arg \min_{\Lambda} \left[ \min_{k} \left( x - \mu_i \right)^{L} \left( x - \mu_i \right)^{\prime} \right]
\]

(4.2)

Now we can apply vector quantization algorithm. Parameters \( \mu_i \) are obtained as centroids, \( c_i \) as the quotient of vector count in cluster \( k \) and total number of vectors in \( X \). Diagonal matrix \( U \) is calculated as inverse matrix of covariance matrix of vectors in cluster \( k \). Transition probability coefficients are set to such a value that the expected state duration is equal to average calculated out of observation sequences [2]

\[
a_{ij} = 1 - \frac{1}{\bar{r}_j}
\]

(4.3)

where \( \bar{r}_j \) is the average duration of state \( j \). These are, of course, only initial values, which are further modified during the corrective training.

**Corrective training**

Basic idea of the corrective training is to improve distinction of the similar models. In order to achieve this it is necessary to:

- choose appropriate goal function,
- implement an algorithm which will approach the maximum of the chosen function in acceptable period of time.

There are several goal functions proposed in literature. One of them is already mentioned and is based on the ML criteria. Let us denote with \( Y \) the set of all feature vector sequences from the training speech database, with \( \Lambda \) the set of parameters of all models, and \( \Lambda_c \) the correct model of the given word. According to ML criteria, \( \Lambda \) is obtained as:

\[
\Lambda = \arg \max_{\Lambda} P(Y \mid \Lambda_c)
\]

(4.4)

In this approach every model’s parameters depend only on sequences which represent that model, and no discriminative training is conducted. Drawbacks in this approach are partially removed by the Maximum Mutual Information (MMI) criteria, when the HMM parameters are obtained by:

\[
\Lambda_{MMI} = \arg \max_{\Lambda} I_{\Lambda} (Y, \Lambda_c)
\]

(4.5)

\[
I_{\Lambda} (Y, \Lambda_c) = \log P(Y \mid \Lambda_c) - \log \sum_{\Lambda} P(Y \mid \Lambda) P(\Lambda)
\]

(4.6)

Each model is now trained not only positively on correct sequences, but also negatively on incorrect ones. Such a goal function leads to increase of model discrimination and better performance, but improvements are possible. Namely this function similarly increases the difference between the models which are already sufficiently distinguished, and those which are critical, i.e. close. That is why we decided to modify this criteria. Like in [1] we adopted the following goal function:

\[
g(\Lambda) = -\sum_{i=0}^{N} \sum_{j=0}^{s} \sum_{M} a_{ij} \log P_0(Y_0 \mid \Lambda_i) - \log P_1(Y_1 \mid \Lambda_i)
\]

(4.7)

where \( Y_0 \) stands for the \( m \)th sequence of the word \( i \), \( M \) is the number of sequences of the word \( i \) in the training set, and \( s \) so called stiffness factor which serves to give more or less weight to critical pairs. It can be seen that the pairs which are already sufficiently distinguished have very small influence on the goal function value in opposition to those “tight” ones or even unrecognized (logarithm difference negative).

In order to achieve the maximum of the goal function, a stochastic method called Evolution Strategies Algorithm ES(1+1) was used. This algorithm adds a random vector of some distribution to the initial parameters. Gaussian distribution for the random vector is adopted. After that the goal function is calculated, and if the new value is greater than initial then we continue the process with the new set of parameters, otherwise we go back to the old ones. Random vector dispersion should be managed in such a way that number of succesful and unsuccessful iterations relates around 5 to 1 [7]. It can be shown that this will be satisfied if in the case of success (greater value) we multiply random vector dispersion with 2, and with 0.8 otherwise.

Straightforward (unoptimized) calculation of the goal function would imply applying each model to each observation sequence. Depending on the number of models, observation sequences and computer power this could take from several minutes up to an hour, for only one iteration. For acceptable solution it is necessary to conduct several hundreds of thousands of iterations. It is obvious that this can’t work without optimization.
Optimization is possible and works fine. Basic assumption is that the segmentations won't be significantly changed if we vary parameters a little, so we keep the segmentation fixed during one set of iterations. This acceptable assumption enables drastic speed up of the algorithm. Namely, each model accumulates certain metric along the whole observation sequence. Each state lasts several frames, so we can also talk about accumulated metric per state. By varying only one state's parameters, we change only contributions of that state to total metrics, since the segmentation didn't change. In this way we reduced the number of parameters which should be simultaneously changed to several hundred, and new goal function value is calculated in several milliseconds (since we calculate only the changed values). After this separate variations of all states in all models we enter the new segmentation and new set of iterations. This method can give acceptable results within several hours of training.

Depending on chosen parameters and training set size this method can, in relatively short period, achieve recognition percent of 100 in the training set, with significant distinction between models. But if training set isn't large enough (and it never is), this will lead to so called overfitting phenomenon. This phenomenon is characterized by excessive model dependence on the training set and its reduced efficiency on some other set. That is why corrective training should be aborted when system performance on an independent set begins to descend.

V WORD SEQUENCE MODEL STRUCTURE
Word sequence is modeled as an array of words and non-speech intervals. So called non-speech state is established for non-speech intervals description. The whole vocabulary is represented by a large trellis in which we have several constraints:

- each sequence must begin with non-speech state,
- non-speech state duration is not limited,
- from the non-speech state it is possible to enter only the beginning state of any word,
- word model has a linear structure and limited states duration (from both sides),
- from the last state of each word it is possible to enter only non-speech state,
- each sequence ends with a non-speech state.

For trellis search we used Viterbi algorithm which observes mentioned constraints. Depending of the desired application, Viterbi search can be "forced" through previously determined sequence of words, which is achieved by creating appropriate trellis.

Besides non-speech state, which does not belong to any word, it is possible to establish some other "words" which could model different kinds of noises. Deploying noises to non-speech state results in an increased number of word insertions, because not all the noises and silence can be modeled with only one state. Some experiments were conducted with one additional "word", and significant improvement was achieved.

VI EXPERIMENT RESULTS
All experiments were conducted on the records of telephone quality. Testing base contained around 3500 records of both isolated and connected digits. Most of the tests were conducted with a reduced training set, because of the time required for system training. Connected words were pronounced fairly quick in often adverse conditions, so it was to expect that the recognition rate would have been even better on some slower pronounced test set.

The aim of the first set of tests was to determine how the system behaves after certain number of corrective training (CT) iterations. We also observed the influence of the number of parameters on the overfitting effect. There were 3 groups of parameters: in the first one we had 2 HMM states per phone (spp) and 6 Gaussian mixtures (gm), in the second 2spp and 3 gm, and in the third 1 spp and 3 gm.

Graph 6.1: Recognition accuracy depending on the CT iterations and parameter number

As expected, overfitting effect is in direct correlation with the number of parameters, more accurately with the quotient of the number of parameters and training set size. When the number of parameters is sufficiently large (first case), after certain number of iterations (concretely 8) we have system degradation on the independent set. Then we should abort the training.

The second test compares system performance with and without additional word which models background noise (Fig.6.2.). On the x-axis we have: the number of false recognitions (fr), insertions (i), deletions (d) and error rate (er) per digit. As expected the most persuasive advance is achieved with the insertions problem, although the number of false recognitions is also significantly reduced. A little unexpected is the deleted digits increase, but nevertheless the overall error rate is diminished.

The third test compares system performance when the distribution function is calculated as a sum of pondered Gaussian mixtures (spg), and when it is calculated by the "winner take all" method (wta) (Fig.6.3.).
The difference in error rate per digit is less than 1% which fully justifies usage of the "winner take all" method, since the gained time savings can be used in another way in order to increase accuracy.

In the fourth test we compared two mentioned methods of dynamic features calculation (Fig.6.4.). In the first case the features are calculated as difference of the two neighbouring frames and filtered afterwards (D), and in the second they are calculated as regression coefficients by (2.1) in window of 70 ms (R 70), and in third also by (2.1) but in the window of 90 ms (R 90).

Regression coefficients method surpassed the old method by more than 2%. We can also see that there is no need to extend window duration more than 70 ms.

The fifth test explored efficiency and significance of preemphasis when the telephone channel is used (Fig.6.2.). We tried with three filters:

- High pass: $s_p(n) = s(n) - 0.7s(n-1)$
- No preemphasis: $s_p(n) = s(n)$
- Low pass: $s_p(n) = 0.2s(n) + 0.6s(n-1) + 0.2s(n-2)$

Test showed that the usage of the classic preemphasis (6.1) in telephone channel was completely unjustified and contributes only to performance degradation. Low pass filter application also degrades performance in smaller amount, so we decided not to use any form of preemphasis in the case of telephone speech.

New system was also tested on the base of isolated words in order to be compared to the old system which didn't use manually labeled phonemes. With all the other parameters fixed, accuracy increased more than 1% (old 98.2%, new 99.3%).

**VII CONCLUSION**

In this paper authors suggested some new and explored some old methods in the process of speech recognition. Some old methods efficiency was confirmed, and some were disputed (for the concrete application). New methods proved very good and significantly improved system accuracy and problem complexity (speed) compared to some conventional techniques proposed in literature. Manual segmentation and corrective training by the mentioned goal function proved extremely good, while significant improvement was gained by inserting additional word for background noise modeling. "Winner take all" method did not contribute to the system performance, but it significantly accelerated and simplified the process of recognition, with negligible loss of accuracy.

**LITERATURE**


**Abstract:** In this paper a system for connected words recognition is presented, based on Hidden Markov Models (HMM). Basic characteristics of this system are: 1) manual segmentation of a part of the speech database, 2) usage of specific criteria (goal function) in the training process, 3) corrective training procedure, 4) usage of the "winner take all" approximation for acoustic vector distribution modeling, 5) usage of the "garbage models" i.e. "words" whose only purpose is to model some form of noise. System is used for both discrete and connected digits recognition in Serbian language. The paper also includes some experiments performed in order to give an estimate of the improvement gained by each of the mentioned methods.

**CONNECTED WORDS RECOGNITION, Darko Pekar, Radovan Obradovic Vlado Deliæ**