Experimental specifics of using HMM in isolated word speech recognition (*)

Dimo Dimov and Ivan Azmanov

Abstract: *The paper presents the authors experience with HMMs (Hidden Markov Models) used for isolated word speech recognition for Bulgarian. Two methods provoked by experiments are discussed, namely for the precise quantization of Gaussian probability distribution/density function (G-pdf) modeling the HMM states' output, and a method for averaging a set of HMMs trained for different versions of a given word.*

Key words: *speech recognition, Hidden Markov Models, Gaussian pdf, speech cepstrum.*

1. INTRODUCTION

Speech Recognition is one of the most dynamic areas of today's Informatics. Speech Recognition could be helpful in many different areas of everyday life – voice control of household appliances, dialing telephone numbers by digits pronunciation, voice navigation helping the driver and so on.

Early achievements in the topic are done still in the 50's by BELL Laboratories and MIT Lincoln Laboratories. A great success has been made in the end of 70's in the area of isolated word recognition, namely by recognition of frequency or cepstral templates [2, 4], using Bayesian estimators, Winner filtering, linear prediction coding, etc., [9]. In the 80's, these techniques were gradually supplanted by HMM based statistical methods [1, 5, 6, 8].

In brief, HMM-s are used for modeling of word(s) pronounced by one or more speakers, where modeling means obtaining a HMM to give a maximum resemblance probability only for the input word it was trained for. So, the recognition of isolated words makes a choice in a set of HMM-s for the HMM best molding given input word [1, 5, 6, 8].

In this paper, practical specifics in setting up of HMM for isolated word recognition are discussed. Two methods are proposed: (1) for adaptive quantization of probabilities molded by Gaussian distributions and (2) a method for averaging a set of HMM-s.

2. FORMULATION

The appropriate conventional symbolism [1, 2, 3, 5, 6, 8, 9] has been obeyed, as far as possible hereinafter.

2.1. Cepstrum of input speech signal

After preprocessing for a noise reduction the input speech signal is extracted by its (most) informative features, i.e. representing it by both enough of accuracy and statistical independency as much as possible. In Speech Recognition, this role plays usually the so-called cepstrum that, in broad terms, carries out the information for energy change of the speech signal [2, 4].

The input signal is split into frames of equal length. For each frame $F_t, t = 1 \div T$, a cepstral vector o_t is calculated that is a Q-dimensional feature vector $o_t = (o_t(i), i = 1 \div Q)$. Thus, the input speech signal is represented by its time sequence $\boldsymbol{O} = (o_1, o_2, ...o_T)$ of cepstral vectors. In practice, Q is chosen in the range of 20÷40 [2, 8].

On Fig.1, a speech cepstrum is illustrated, where the darker areas correspond to words or syllables, i.e. to higher energy of the signal.

We generally assume that the input speech is already split into isolated words, or respective syllables, or other phonetic-lexical units, e.g. the "allophones" from [7], which cepstral representation $\boldsymbol{O}$ is to be given to a HMM for recognition.

(*) This work was supported by following grants of the Institute of Information Technologies at Bulg. Academy of Sciences (BAS): Grant #010056/2003 of BAS, Grant # I-1306/2003 of the National Science Fund at Bulg. Min. of Education & Science, and Grant # RC6/2004 of the CICT Development Agency at Bulg. Min. of Transport & Communication.

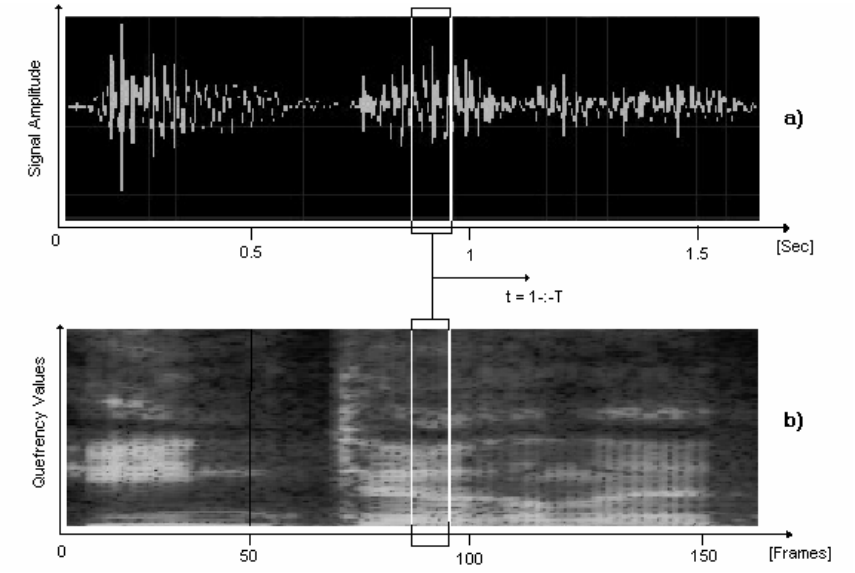

Fig.1. (a) An original signal of the word "*internet*", and (b) its cepstrum

2.2. HMM – Hidden Markov Models

Like many other recognition structures, HMM-s are also foreseen to work in two different regimes: (1) training and (2) recognition.

When a sequence $\boldsymbol{O} = (o_1, o_2, ...o_T)$, in our case consisting of cepstral vectors, feeds the HMM input, the conditional probability $P(\boldsymbol{O} \mid \boldsymbol{M})$ is expected at the HMM output. Here $\boldsymbol{M}$ marks the training (setting up) degree of HMM respecting $\boldsymbol{O}$. Given HMM is considered trained for the word $\boldsymbol{O}$, if the output $P(\boldsymbol{O} \mid \boldsymbol{M})$ reaches maximum at $\boldsymbol{O}$. We assume that the elements $o_t, t = 1 \div T$ of $\boldsymbol{O}$ are simultaneously passed to the HMM in each regime, training or recognition, i.e. the input space can be considered TQ dimensional one, see Fig.2.

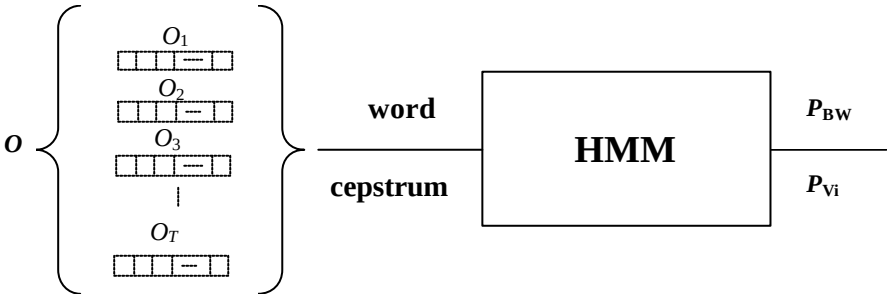

Fig.2. For each input word $\boldsymbol{O}$ the HMM generates a corresponding probability $P(\boldsymbol{O} \mid \boldsymbol{M})$ that could be computed either by Baum-Welch or by Viterbi algorithms

Given HMM could be defined by the five (S, A, s_0, O, B), where:

- $S = \{s_1, s_2, ..., s_N\}$ is the set of the internal states;
- $A = [a_{ij}]$ is the matrix of transitions a_{ij} (conditional probabilities for transition) from state s_i to state s_j, $a_{ij} = P(s_j \mid s_i)$, $1 \le i,j \le N$. Obviously $\sum_{j=1}^{N} a_{ij} = 1$, $1 \le i \le N$.
- s_0 is the starting state;
- $O = \{o_1, o_2, ..., o_T\}$ is the set of possible observations;
- $B = [b_{ij}]$ is the matrix of conditional probabilities for given observation the HMM being in some internal state, i.e. $b_{ij} = b_i(o_j) = P(o_j \mid s_i)$, $1 \le j \le T$, $1 \le i \le N$.

Thus, HMM is considered a probability finite automaton, whose internal states (in contrast to classical Markov models) cannot be directly measured (observed) but only indirectly.

In speech recognition, the most spread HMM-s are the so called "left-right schemes" also known as "Bakis machines" [3, 6]. Here, instead of the complete transition graph, the matrix A reflects the streamness of modeled speech signal in the time, i.e. the spread direction is "not backward", and the pre-history dependency is minimal (up to 1÷3 states back), [1, 6, 8], see Fig.3.

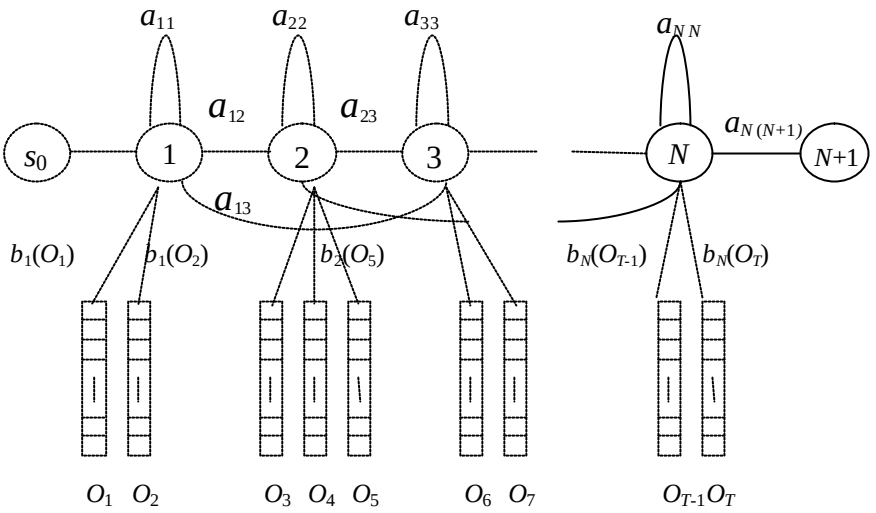

Fig.3. A left-right HMM used in speech recognition

The observation set $\boldsymbol{O}$ as numbered above strictly corresponds to the time sequence $\boldsymbol{O} = (o_1, o_2, ..., o_T)$ of cepstral vectors at the HMM input. The way $\boldsymbol{M}$ of HMM-training consists either of choosing N towards T, (i.e. $N<T$ or $N=T$ or $N>T$), or of calculating the transition matrix A as well as the matrix B of the output probabilities. It is mostly realized by an iterative optimization algorithm of Baum-Welch, while the real recognition – by the simpler and less exact approach of Viterbi.

Thus, HMM calculates the output $P(\boldsymbol{O} \mid \boldsymbol{M})$ during the internal time $t, 1 \le t \le T$, as were reproducing the complete input $\boldsymbol{O}$, state by state, see Fig.3. On the other hand, from external viewpoint, HMM operates as a black box, in time intervals of length T that can vary at different input words, see again Fig.2 and Fig.1.

2.3. Calculation of the output probability $P(\boldsymbol{O} \mid \boldsymbol{M})$

In each moment $t, 1 \le t \le T$, HMM could be in an arbitrary state $s_i, i = 1 \div (N-1)$, starting from s_0 (in t=0) and finishing in s_N (in t=T+1). In this sense, each vector $o_t, t = 1 \div T$ can correspond to (or be mould by) every internal state $s_i, i = 1 \div (N-1)$, despite the fact that some state(s) would be the most probable one(s) for this o_t. Because of the independence assumption for the state events, $P(\boldsymbol{O} \mid \boldsymbol{M})$ can be calculated as the sum:

$$P(\boldsymbol{O} \mid \boldsymbol{M}) = \sum_{\{X\}} P(\boldsymbol{O}, X \mid \boldsymbol{M}) \ , \qquad (1)$$

over all possible ways $X = (x_t, t = 1 \div T)$, along the states $x_t \in S$ of HMM, from the starting s_0 to the final s_N. This interpretation is often illustrated with a network diagram, cf. Fig.4, and it is easy to check by it that:

$$P(\boldsymbol{O}, X \mid \boldsymbol{M}) = a_{x(0)x(1)} \prod_{t=1}^{T} b_{x(t)}(o_t) a_{x(t)x(t+1)} \ , \qquad (2)$$

where

$a_{x(0)x(1)}=1$, $a_{x(t)x(t+1)}$ and $b_{x(t)}(o_t)$ are the corresponding transition and output probabilities for the given training $\boldsymbol{M}$.

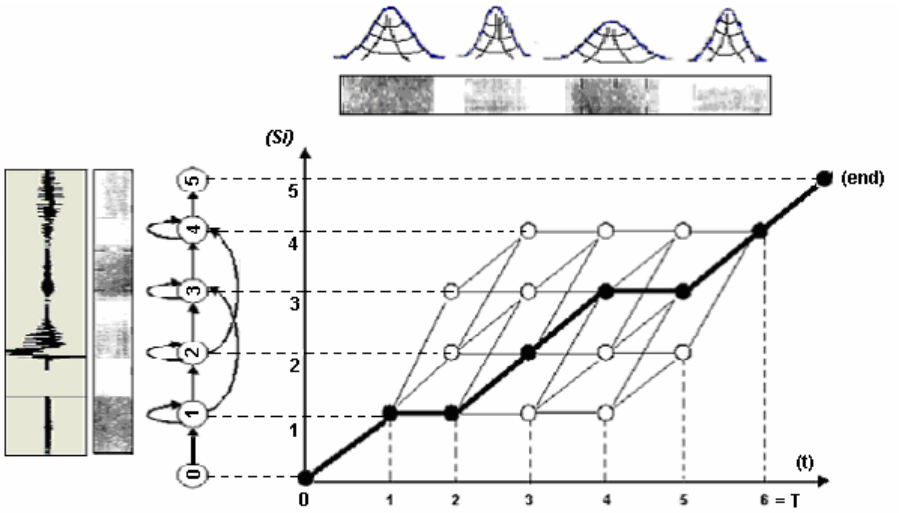

Fig.4. An illustration of HMM progress in the time, where the possible directions are to the right (at time $t=1\div T$) and/or to the top (over the states s_i, $i=1\div N$)

In this network interpretation, HMM being in given state s_i, at the moment t, as if "generates" the corresponding input cepstral vector o_t, with a probability

$$P(s_i(t)\,|\,\boldsymbol{M}) = a_i(t\,|M)b_i(o_t)b_i(t\,|\,\boldsymbol{M}) \ , \qquad (3)$$

where $a_i(t\,|\,\boldsymbol{M})$ and $b_i(t\,|\,\boldsymbol{M})$ are the corresponding complete probabilities, "forward to s_i" and "backward till s_i", associated with any state s_i, $i=1\div(N-1)$, at the moment t, $t=1\div T$:

$$a_i(t\,|\,\boldsymbol{M}) = \sum_{\vec{X}\in\{(0,0)\to(i,t)\}} P(\boldsymbol{O},\vec{X}\,|\,\boldsymbol{M}) = \sum_{\vec{X}\in\{(0,0)\to(i,t)\}} a_{x(0)x(1)}\prod_{t=1}^{t-1} b_{x(t)}(o_t)a_{x(t)x(t+1)} = \sum_{j=1}^{N-1} a_j(t-1\,|\,\boldsymbol{M})a_{ji}b_i(o_t) \qquad (4,4a)$$

$$b_i(t\,|\,\boldsymbol{M}) = \sum_{\vec{X}\in\{(i,t)\to(N,T+1)\}} P(\boldsymbol{O},\vec{X}\,|\,\boldsymbol{M}) = \sum_{\vec{X}\in\{(i,t)\to(N,T+1)\}} a_{i,x(t+1)}\prod_{t=t+1}^{T} b_{x(t)}(o_t)a_{x(t)x(t+1)} = \sum_{j=1}^{N-1} a_{ij}b_j(o_t)b_j(t+1\,|\,\boldsymbol{M})$$

This way the output probability $P(\boldsymbol{O}\,|\,\boldsymbol{M})$ obtains the well known expression:

$$P(\boldsymbol{O}\,|\,\boldsymbol{M}) = \sum_{i=1}^{N-1} P(s_i(t)\,|\,\boldsymbol{M}) = \sum_{i=1}^{N-1} a_i(t\,|\,\boldsymbol{M})b_i(o_t)b_i(t\,|\,\boldsymbol{M}), \quad t=1\div T \ , \qquad (5)$$

that is independent on the time t, cf. (1), and is often replaced by the simpler:

$$P(\boldsymbol{O}\,|\,\boldsymbol{M}) = b_0(0\,|\,\boldsymbol{M}) \ , \quad t=0 \ , \qquad (5a)$$

or $$P(\boldsymbol{O}\,|\,\boldsymbol{M}) = a_N(T+1\,|\,\boldsymbol{M}) , \quad t=T+1 \ . \qquad (5b)$$

Of course, (5a) and/or (5b) do not change the fact that the $P(\boldsymbol{O}\,|\,\boldsymbol{M})$ calculation needs extra calculation of all, $N(T+1)$ in number, probabilities of the type (4,4a). But, it is equation (5) that explicitly shows the difference between both basic methods for $P(\boldsymbol{O}\,|\,\boldsymbol{M})$ calculus, Baum-Welch and Viterbi, as well as which of them to choose for the $\boldsymbol{M}$ training iterations.

2.4. Training and recognition

In a HMM based recognition system, a separate HMM is build (trained) for each word $\boldsymbol{O}, \boldsymbol{O}\in\boldsymbol{W}$, $\boldsymbol{W}$ the set under recognition. The training of each HMM consists in setting up (adjusting calculus) of its matrices A and B, so that the probability $P(\boldsymbol{O}\,|\,\boldsymbol{M})$ of "its own" word $\boldsymbol{O}\in\boldsymbol{W}$ to be maximal and as high as possible. Baum proposed a monotone converging algorithm for recursive adjustment of HMM to the optimal $\boldsymbol{M}_{\mathrm{BW}}$ training, so that

$P(\boldsymbol{O}\,|\,\boldsymbol{M}_{\mathrm{BW}})$, defined with (5), and first proposed by Welch [6], to reach the maximum in $\boldsymbol{O}$, i.e.: $\boldsymbol{P}_{\mathbf{BW}} = P(\boldsymbol{O}\,|\,\boldsymbol{M}_{\mathrm{BW}}) = \max_{\boldsymbol{M}}\{P(\boldsymbol{O}\,|\,\boldsymbol{M})\}$, (6)

The Baum-Welch computation schema and many modifications as well [1, 3, 5, 6, 8], could be also used for almost optimal calculus of $P(\boldsymbol{O}\,|\,\boldsymbol{M}_{\mathrm{BW}})$, namely by a formulae similar to (1) and inherited from Viterbi's approach well-known in the graphs' theory:

$$P^*(\boldsymbol{O}\,|\,\boldsymbol{M}) = \max_{\{X\}} P(\boldsymbol{O}, X\,|\,\boldsymbol{M}) \;<\; \sum_{\{X\}} P(\boldsymbol{O}, X\,|\,\boldsymbol{M}) = P(\boldsymbol{O}\,|\,\boldsymbol{M}) \;, \tag{7}$$

and consequently:

$$\boldsymbol{P}_{\mathbf{Vi}} = P(\boldsymbol{O}\,|\,\boldsymbol{M}_{\mathrm{Vi}}) = \max_{M}\{P^*(\boldsymbol{O}\,|\,\boldsymbol{M})\} \;<\; \boldsymbol{P}_{\mathbf{BW}} \;. \tag{7a}$$

Here, instead of summation over all possible paths, cf. (1), $P(\boldsymbol{O}\,|\,\boldsymbol{M}_{\mathrm{Vi}})$ is evaluated only over the maximal path (it is shown in black on Fig.4). Similar replacements should also be done defining respectively the new $a_i(t\,|\,\boldsymbol{M})$ and $b_i(t\,|\,\boldsymbol{M})$, by analogy to (4,4a), what usually sharply improves the computational performance.

Viterbi's algorithm is well appropriate for the recognition regime, when HMM is trained enough. In the very beginning of the training and often during whole training, Baum-Welch algorithm is totally recommended, especially if it is important to ensure the monotony convergence conditions, i.e. without freezing in any false optimums. And the last could be usable, e.g. when appropriate correction in the number of HMM states is aimed, simultaneously with training, for reaching the optimal correspondence $N \Leftrightarrow T$.

2.5. Transition matrix modeling

Often for the $P(\boldsymbol{O}\,|\,\boldsymbol{M})$ calculation [1, 6, 8], the output probabilities $b_i(o_t)$, i.e. the B matrix elements for given state s_i of HMM, are modeled by a Q-dimensional Gaussian distribution of density $G(m_i,\Sigma_i)$, $i = 1 \div (N-1)$. The centre m_i presents the mean of all cepstral vectors o_t the HMM "generates" being in the state s_i. The covariance matrix Σ_i is most often assumed diagonal one, i.e. that all cepstral vector components $o_t \in \boldsymbol{O}$, $o_t(q)$, $q = 1 \div Q$, are uncorrelated. Thus, each model $G(m_i,\Sigma_i)$ is independently examined along its coordinates $(q), q = 1 \div Q$, i.e. $b_i(o_t)$, and can be modeled by the multiplication:

$$b_i(o_t) = G(o_t; m_i, s_i) = \prod_{q=1}^{Q} G(o_t; m_{iq}, s_{iq}) = \frac{1}{\sqrt{(2p)^Q}} \prod_{q=1}^{Q} \frac{1}{s_{iq}} \exp\left(-\frac{(o_t - m_{iq})^2}{2s_{iq}^2}\right), \tag{8}$$

$$m_i = (m_{iq}\,|\,q = 1 \div Q)\;,\; s_i = (s_{iq}\,|\,q = 1 \div Q)\;,\; i = 1 \div (N-1),\; t = 1 \div T \quad ,$$

where s_i are the vectors of corresponding mean-square deviations.

Obviously, $P(\boldsymbol{O}\,|\,\boldsymbol{M})$ could overcome "very quickly" the lower limit of the computer numbers representation, e.g. ~1.E-308 for the C-type "double". For instance, in accordance with (3), (4,4a), (5) and (8), we can evaluate roughly for the output $P(\boldsymbol{O}\,|\,\boldsymbol{M})$:

$$P(\boldsymbol{O}\,|\,\boldsymbol{M}) \sim \left(\frac{\Delta}{6s}\right)^{QNT} \left(\frac{1}{TN}\right)^{T} < 1 \;. \tag{8a}$$

As better a HMM is trained for given word, as "sharper" its Gaussians become that results in further more diminish of the modeled probabilities. For this reason a computation scaling for the intermediate probabilities $a_i(t)$ and $b_i(t)$ is usually proposed, [6]. This way the whole computation range is pursued. But our experiments show that often no scaling can solve the problems either with underflow or with extra overflows appearing as well. A solution can be in using of log-probabilities, where it is appropriate, but the more serious approach lies in an adaptive and more precise calculation using the modeling Gaussians.

3. ADAPTIVE AND PRECISE QUANTIZATION OF THE MODELING GAUSSIANS

Figuratively, each state of the HMM moulds the input sequence $\boldsymbol{O} = (o_1, o_2, ..., o_T)$ by a combination of N-1, in their number, Q-dimensional "Gaussians" each of them being just a multiplication of Q one dimensional (1D) Gaussians.

By definition, the density $G(o; m, s)$, $m = m_{iq}, s = s_{iq}$, $i = 1 \div (N-1)$, $q = 1 \div Q$, of each 1D-Gaussian is a continuous function (pdf) over the input $o = o_{iq}$ and the parameters μ and s. This leads to a necessity of suitable quantization. Mainly, by reasons of uniformity, i.e. of isometrics for each input cepstrum element, we assume that the interval Δ of quantization is equal along each coordinate of all the considered Gaussians.

As known the G-pdf as density is a differential of the G-pdf as distribution, so the probability $P(x)$, modeled by given 1D-Gaussian $G(x) = G(x; m, s)$ could be calculated by the well known "trapezium-like" formulae:

$$P(x) = P(x - \Delta/2 \le x < x + \Delta/2) = \tfrac{1}{2}\Delta\big(G(x - \Delta/2) + G(x + \Delta/2)\big). \qquad (9)$$

The probability $P(x)$ could be approximated more precisely, but the problem of Δ preliminary choice is more important in the case. Obviously, if Δ decreases, the accuracy of $P(x)$ calculation is increasing (see Fig.5), while the $P(x)$ decreases itself that could lead to undesirable "underflow" in the computer calculations.

On the other hand, by a chosen discrete Δ the computation accuracy remains depending only on the value of s, the mean-square deviation of the given 1D-Gaussian that can vary with each step of HMM training algorithm. Obviously, there exists a boundary value s_0, $s_0 = s_0(\Delta)$, depending on the choice of Δ only, so that for each s, $s < s_0$, the calculation of $P(x)$, e.g. by (9), will lose an accuracy. An illustration of this is given on Fig.5.

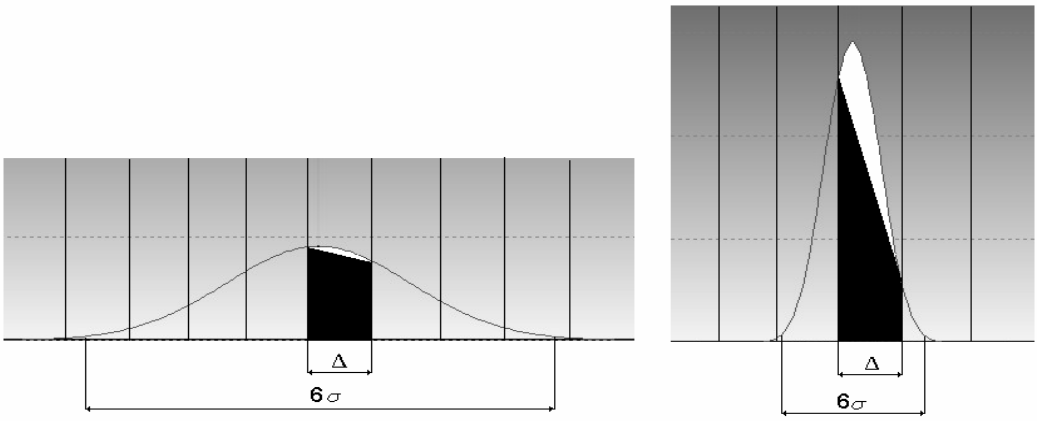

Fig.5. Computational precision of probabilities depending on the choice of s and Δ: (a) by $s \ge s_0(\Delta)$; (b) by $s < s_0(\Delta)$

So, the formulae (9), we call it "*large model*", is used for 1D-Gaussians of $s \ge s_0$. In the cases of $s < s_0$, we introduce another calculation, called here "*slim model*", where the classical integration formulae is used:

$$P(X) = \int_{X-\Delta/2}^{X+\Delta/2} G(x, m, s)\,dx = \Phi\left(\frac{X - m + \Delta/2}{s}\right) - \Phi\left(\frac{X - m - \Delta/2}{s}\right). \qquad (10)$$

Here, $\Phi(X)$ is defined like the well-known Laplace function:

$$\Phi(X) = \frac{1}{\sqrt{2p}} \int_{-\infty}^{X} \exp(-x^2/2)\,dx = \int_{-\infty}^{X} G(x; 0, 1)\,dx \qquad (10a)$$

and could be tabulated only once at the beginning.

The threshold s_0 could be evaluated very precisely mainly considering the calculation error equality at the transition between both formulas. For the present experiments, we choose as best the values: $\Delta=1$ and $s_0=1$.

4. AN AVERAGED HMM FOR GIVEN WORD VERSIONS

So far we have considered the training of HMM for given word only. But we often have to deal with word versions caused by differences of the speaking speed, speaker's timbre, dialect, etc. Then we need a generalization for the set of resulting HMMs to recognize as many versions as possible and with enough of confidence. We have examined two possible approaches:

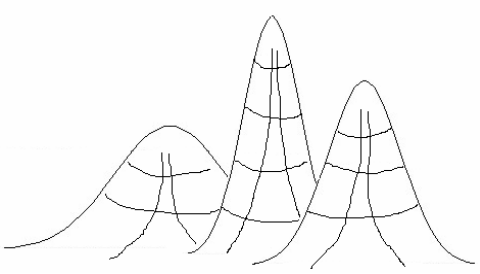

Fig. 6. A Gaussian mixture

1. Instead of using a unique (Q-dimensional) Gaussian, each HMM state could be modeled by mixture of Gaussians (see Fig.6), each one reflecting the version specifics of the given word in the corresponding (to the HMM state) interval of pronunciation. The novel HMM has to be "uniformly" trained with all versions of the word at each training step.

2. Similarly to the first approach, but the Gaussian mixture to be formed after training of all HMM-s for the given word versions. Besides, a new Gaussian can be also defined as an average of this (Q-dimensional) Gaussian mixture.

The first approach seems more precise, but it requires much longer time for training, almost in a square degree longer because of each 1D-Gaussian of the mixture as well as each version of the word.

We have preferred the second approach mostly considering its performance effectiveness. Thus, each extra version of the given word is molded more simply, by training of its own HMM only followed by a final actualization of the respective averaged Gaussians. This way we reach to the idea of an averaged HMM, where the necessary restriction for constant (a priori given) number N of the HMM internal states seems very acceptable.

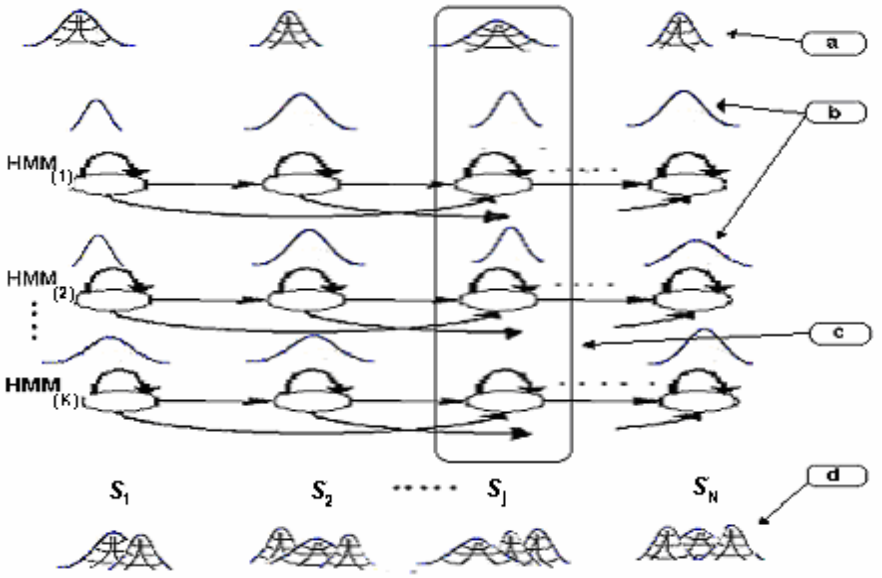

Fig.7. HMM averaging: (a) Q-dimensional Gaussians for the states of $HMM_{(1)}$, (b) corresponding 1D-Gaussians over the axis (q), $q=1\div Q$) for $HMM_{(1)}$ and $HMM_{(2)}$, (c) we average all 1D-Gaussians over the axis (q) for the state s_i of all $HMM_{(1\div K)}$, (d) Mixture of Q-dimensional Gaussians for the states of the averaged $HMM_{(*)}$

Let us have K versions less or more distinguishing the input template (word). Let us also assume possible statistical dependences among the versions' cepstruma only by separate coordinates (q), q=1÷Q. Thus, we can average the corresponding HMM-s sequentially, state by state, coordinate by coordinate, i.e. for given coordinate (q), for the state s_i, i=1÷(N-1), we have K 1D-Gaussians to average, see Fig.7.

The centre of the averaged Gaussian is evaluated as the mean vector:

$$\vec{m} = (\vec{m}_{(1)} + \vec{m}_{(2)} + ... + \vec{m}_{(K)})/K \ , \ \vec{m}_{(k)} = (m_{(k)}^{(1)}, m_{(k)}^{(2)}, ... m_{(k)}^{(Q)}) \ , \ k = 1 \div K \quad . \tag{11}$$

For the vector of mean-square deviations s (or dispersions s^2) we have:

$$K^2 s^2 = (s_{(1)}^2 + s_{(2)}^2 + ... + s_{(K)}^2) + 2 \sum_{1 \le i < j \le K} C_{(i,j)} \quad , \tag{12}$$

where $C_{(i,j)} = \mathrm{E}[(o_i - m_i)(o_j - m_j)]$, $s_{(i)}^2 = \mathrm{E}[(o_i - m_i)^2] = C_{(i,i)}$, $i, j = 1 \div (N-1)$, $\mathrm{E}(.)$ is the mean operator, and $C_{(i,j)}$ are the so called co-variation moments. Consequently:

$$(s_{(1)}^2 + s_{(2)}^2 + ... + s_{(K)}^2) \le K^2 s^2 \le (s_{(1)} + s_{(2)} + ... + s_{(K)})^2 \quad , \tag{12a}$$

where the left equality is reached for independent items (1D-Gaussians), while the right equality – for fully dependent ones. The inequalities (12a) give us the interval of possible experimental variation of mean-square deviations of the averaged Q-D-Gaussian.

The averaged transition matrix is calculated item by item, by the incompatibility assumption for corresponding probabilities, i.e.:

$$a_{ij} = (a_{ij}^{(1)} + a_{ij}^{(2)} + ... + a_{ij}^{(K)})/K \quad , \quad i, j = 1 \div (N-1) \ . \tag{13}$$

It is experimentally concluded that the transition matrix averaging is better to be performed mean-geometrically instead of mean-arithmetically if following (13), because of recognition improvements in this way.

5. EXPERIMENTAL RESULTS AND ANALYSIS

The experiments are done on an IBM PC compatible: Intel Pentium4 3GHz CPU, 1MB L2-cache, 512MB RAM, 160GB HDD. The operational systems is Windows XP(SP2).

The experimental program operates with standard WAV-files recorded on 22 kHz, 16 bits per datum, mono signal.

The experiment database consists of about 30 words, about 5 versions per word. For each version a separate HMM is trained. For the homonymous word versions an averaged HMM has been computed using the above proposed method. By no special measures of optimizing the computing environment, the average training time for the whole database is about 5 minutes, i.e. about 2 sec per word version.

Some fixed parameters are: (a) the frame F length Δt for windowing the cepstrum computing, Δt=10ms, (b) the cepstral vectors' length Q=40, (c) the HMM states' number, N=7, (d) the maximal number of training iterations =15. Experimental timing for 5 words of the database is given in the following table:

Input word spelling	Frames' count (10ms per frame)	Preprocessing [sec]	Baum-Welch training [sec] & [sec/frame]	Baum-Welch recogn. [sec]	Viterbi recogn. [sec]
delete	185	0.016	1.187 / 0.079	0.500	0.422
folder	115	0.013	0.828 / 0.055	0.375	0.351
internet	153	0.014	0.719 / 0.048	0.484	0.465
menu	107	0.012	0.688 / 0.046	0.469	0.443
open	167	0.015	0.875 / 0.059	0.406	0.391

Each word versions are preliminary processed, i.e. the sound noise is cleared as better as possible and each word is isolated. In such conditions the average recognition rate is about 98%. For the present the automatic end-point detection is not enough precise that drops down the recognition rate to about 75-80%. However this should be considered

optimistic. Experiments have been also made for words pronounced by different speakers, where the recognition rate decreases to about 50-60%. We analyze that the recognition rate is to become better if train the averaged HMM by more versions per word, what, of course will increase the training duration.

6. CONCLUSIONS AND FUTURE WORK

Experimentally provoked, two methods have been proposed for improvements in the HMM based speech recognition, namely:

1. A method for adaptive and precise computation of probabilities modeled by Gaussian pdf-s. The method aims an environment providing for monotone convergence of the HMM training by Baum-Welch and the consecutive recognition by Viterbi.

2. A method for HMM averaging that combines separate HMM-s already trained for different versions of words under recognition. The method aims more effective implementation of the Gaussian mixture idea, i.e. for mixing/averaging finally instead of at the each iteration of HMM training as it is popular now.

In near future we intend to develop an approach for optimal choice of the HMM internal states number generally depending on the input word lengths and contents.

Final aim of this research is to develop a HMM based method for speech recognition with automatic setting up to the specifics of Bulgarian language. A definite hope in this respect is voted to the experimental hypothesis of [7] for the almost fully representation completeness of a set of about 500÷1500 Bulgarian allophones.

REFERENCES

[1] Cernocky J., Hidden Markov Models - an Introduction, Brno University of Technology, Faculty of Information Technology,13p., (http://www.fit.vutbr.cz/~cernocky/oldspeech/lectures/hmm.pdf)

[2] Dunn B., Speech Signal Processing and Speech Recognition, Current Topics in DSP, Speech Proc.2, RBD, May 13, 2003, 34p., (http://www.caip.rutgers.edu/~rabinkin/DSP_no_audio.pdf).

[3] Gilloux M., Hidden Markov Models in Handwriting Recognition. *Fundamentals in Handwriting Recognition*, in S. Impedovo (ed.) NATO ASI Series, Series "F": Computer and System Sciences, Vol. 124, Berlin: Springer-Verlag, 1994, pp.264-288.

[4] Jackson P., Advanced Digital Signal Processing, UniS, Univ. of Surrey, 14p., (http://www.ee.surrey.ac.uk/CE/technical/advdsp.html).

[5] Medvedev I., Isolated-Word Speech Recognition Using Hidden Markov Models, MIT, April 19, 2001, MA, US, 9p., (http://www.cnel.ufl.edu/~yadu/report1.html).

[6] Rabiner L. R., Tutorial on Hidden Markov Models and Selected Applications in Speech Recognition, *Proceeding of the IEEE*, Vol.77, No.2, (1989), pp.257-286.

[7] Totkov G. A., Conceptual and Computer Modelling of Language Structures and Processes (with Applications for the Bulgarian Language), D.Sc. dissertation, Spec. #01-01-12 "Informatics", Univ. of Plovdiv, 2004, (by resume in Bulgarian, 57p.).

[8] Wu Y., A. Ganapathiraju, and J. Picone, Baum-Welch reestimation of Hidden Markov Models, Report of Inst. for Signal and Information Processing, June 15, 1999, Mississippi State Univ., MI, US, (http://ieeexplore.ieee.org/iel5/5/698/00018626.pdf).

[9] Vaseghi S. V., Advanced Digital Signal Processing and Noise Reduction, (2d ed.), John Wiley & Sons, Ltd., NY, 2000, pp.227-261.

ABOUT THE AUTHORS

Dimo Dimov, Assoc. Prof., Ph.D., Institute of Information Technologies, Bulgarian Academy of Sciences, Tel: (+359 2) 8706493, E-mail: dtdim@iinf.bas.bg

Ivan Azmanov, Ph.D. student, Institute of Information Technologies, Bulgarian Academy of Sciences, Tel: (+359 2) 8706493, E-mail: ivanazmanov@iinf.bas.bg