BULGARIAN ACADEMY OF SCIENCES

CYBERNETICS AND INFORMATION TECHNOLOGIES • Volume **7**, No 3

Sofia • 2007

Heuristic Improvements of the HMM Use in Isolated Word Speech Recognition*

Dimo Dimov, Ivan Azmanov

Institute of Information Technologies, 1113 Sofia

E-mail: dtdim@iinf.bas.bg

Abstract: *The paper presents the authors' experience with HMMs (Hidden Markov Models) used for isolated word speech recognition in Bulgarian. Two methods provoked by experiments are discussed, namely: (i) the precise quantization of Gaussian probability density function (G-pdf) modeling the HMM states' output, and (ii) a method for averaging a set of HMMs trained for different versions of a given word. A universal threshold is evaluated for switching between the "large" and "slim" models of G-pdf defined here. Experimental results of the threshold usage for averaging of HMMs are also reported.*

Keywords: *Speech recognition, HMM, 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 of XXth century 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 supplemented by HMM based statistical methods [1, 5, 6, 8].

* This work was partially supported by the following grants: Grant No 010088/2007 of Bulgarian Academy of Sciences, and Grant No B-G-17/2005 of the Bulgarian Ministry of Education & Science.

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. Hence, the recognition of isolated words makes a choice in a set of HMM-s for the HMM best molding the 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: a method for adaptive quantization of probabilities molded by Gaussian distributions, see Sections 3 and 4, and a method for averaging a set of HMM-s, see Section 5. Section 2 describes the necessary background for the problems considered. The conducted experiments are discussed in Section 6, and future work plans are discussed finally in Section 7.

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 an input speech signal

After preprocessing for noise reduction, the input speech signal is extracted by its (most) informative features, i.e. representing it by both enough 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 $O_t = (o_t(q),\ q = 1 \div Q)$. 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].

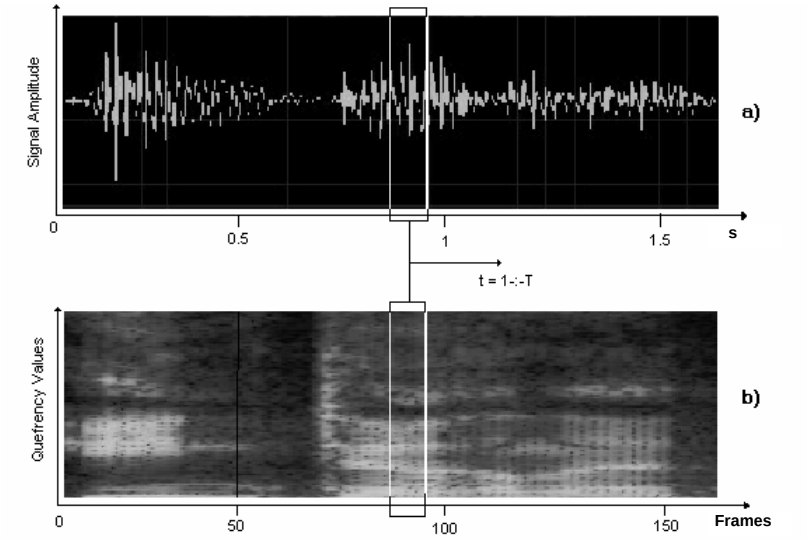


Fig. 1. An original signal of the word "*internet*" (a), and its cepstrum (b)

In 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.

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 (Fig. 2).

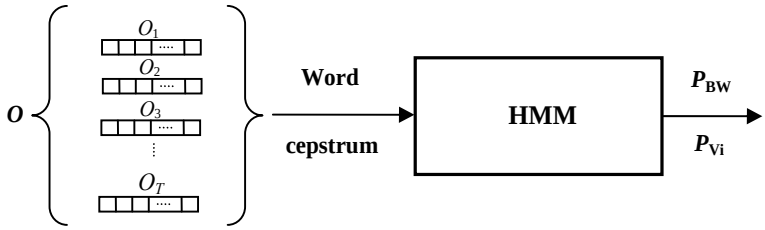


Fig. 2. For each input word $\boldsymbol{O}$ the HMM generates a corresponding probability $P(\boldsymbol{O} \mid \boldsymbol{M})$ that could be computed either by Baum-Welch or by Viterbi algorithms

A given HMM can 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;
- $A = \|a_{i,j}\|_{(N+1)\times(N+1)}$ is the matrix of conditional probabilities $a_{i,j}$ for transition from state s_i to state s_j, $0 \le a_{i,j} = P(s_j \mid s_i) < 1$, $i, j = 1 \div N+1$; obviously, $\sum_{j=1}^{N+1} a_{i,j} = 1$, $i = 1 \div N+1$;
- s_0 is the starting state; we mark with s_{N+1} one more auxiliary state, the finishing one, and set for the respective extra transitions: $0 < a_{N,N+1} < 1$, and $a_{0,1} = a_{N+1,N+1} = 1$;
- $O = \{O_1, O_2, ..., O_T\}$ is the set of possible observations;
- $B = \|b_{i,j}\|_{N\times N}$ is the matrix of conditional probabilities for given observation, the HMM being in some internal state, i.e. $b_{i,j} = b_i(O_j) = P(O_j \mid s_i)$, $j = 1 \div T$, $i = 1 \div N$.

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 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] (Fig. 3).

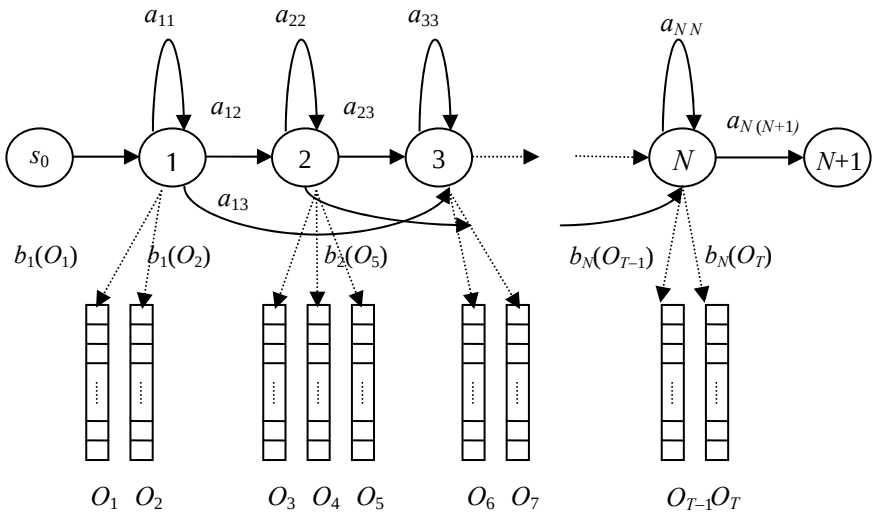


Fig. 3. A left-right HMM used in speech recognition

The observation set 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 if reproducing the complete input $\boldsymbol{O}$, state by state (Fig. 3). On the other hand, from an external viewpoint, HMM operates as a black box, in time intervals of length T that can vary at different input words, see again Figs. 2 and 1.

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

At each moment $t, 1 \le t \le T,$ HMM could be in an arbitrary state s_i, $i = 1 \div N$, starting from s_0 (in t=0) and finishing in s_{N+1} (in $t=T$+1). In this sense, each vector O_t, $t = 1 \div T,$ can correspond to (or be molded by) every internal state s_i, $i = 1 \div N$, despite the fact that some state(s) would be the most probable one(s) for this O_t.

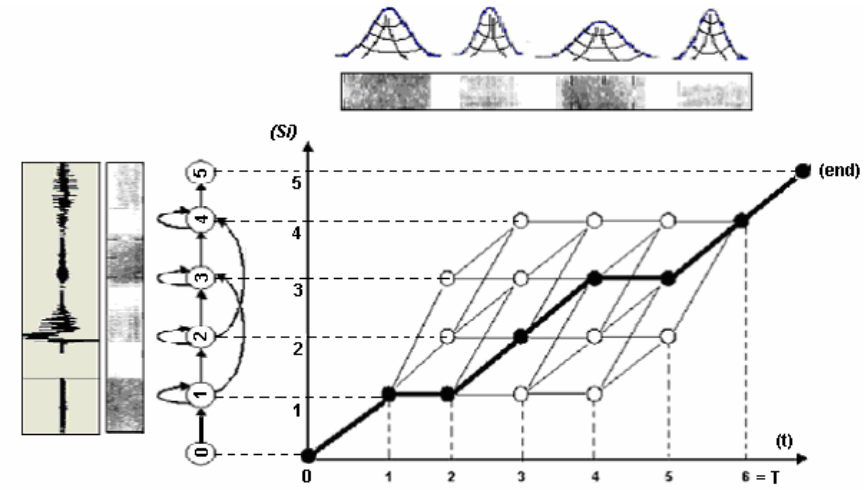


Fig. 4. An illustration of HMM progress in time, where the possible directions are to the right (at time $t = 1 \div T$) and/or to the top (over the states s_i, $i = 1 \div N$)

Because of the independence assumption for the state events, $P(\boldsymbol{O} \mid \boldsymbol{M})$ can be calculated as the sum:

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

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+1}. This interpretation is often illustrated by a network diagram (Fig. 4), and it is easy to check by it that:

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

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 $\boldsymbol{M}$.

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

$$P(s_i(t) \mid \boldsymbol{M}) = \alpha_i(t \mid M) b_i(O_t) \beta_i(t \mid \boldsymbol{M}), \tag{3}$$

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

$$\begin{aligned} \alpha_i(t \mid \boldsymbol{M}) &= \sum_{\overleftarrow{X} \in \{(0,0) \to (i,j)\}} P(\boldsymbol{O}, \overleftarrow{X} \mid \boldsymbol{M}) = \\ &= \sum_{\vec{X} \in \{(0,0) \to (i,t)\}} a_{x(0),x(1)} \prod_{\tau=1}^{t-1} b_{x(\tau)}(O_\tau) a_{x(\tau),x(\tau+1)} = \sum_{j=1}^{N} \alpha_j(t-1 \mid \boldsymbol{M}) a_{j,i} b_i(O_t), \end{aligned} \tag{4}$$

$$\beta_i(t \mid \boldsymbol{M}) = \sum_{\overleftarrow{X} \in \{(i,t)\to(N,T+1)\}} P(\boldsymbol{O}, \overleftarrow{X} \mid \boldsymbol{M}) =$$

(4a)

$$= \sum_{\overleftarrow{X} \in \{(i,t)\to(N,T+1)\}} a_{i,x(t+1)} \prod_{\tau=t+1}^{T} b_{x(\tau)}(O_\tau) a_{x(\tau),x(\tau+1)} = \sum_{j=1}^{N} a_{i,j} b_j(O_t) \beta_j(t+1 \mid \boldsymbol{M}) .$$

In this way the output probability $P(\boldsymbol{O} \mid \boldsymbol{M})$ obtains the well known expression:

(5) $$P(\boldsymbol{O} \mid \boldsymbol{M}) = \sum_{i=1}^{N} P(s_i(t) \mid \boldsymbol{M}) = \sum_{i=1}^{N} \alpha_i(t \mid \boldsymbol{M}) b_i(O_t) \beta_i(t \mid \boldsymbol{M}), \quad t = 1 \div T ,$$

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

(5a) $$P(\boldsymbol{O} \mid \boldsymbol{M}) = \beta_0(0 \mid \boldsymbol{M}), \quad t = 0 , \text{ or}$$

(5b) $$P(\boldsymbol{O} \mid \boldsymbol{M}) = \alpha_{N+1}(T+1 \mid \boldsymbol{M}), \quad t = T+1 .$$

Of course, (5a) and/or (5b) do not change the fact that the $P(\boldsymbol{O} \mid \boldsymbol{M})$ calculation needs extra calculation of all, $(N+2)(T+2)$ in number, probabilities of the type (4), (4a). But it is equation (5) that explicitly shows the difference between both basic methods for $P(\boldsymbol{O} \mid \boldsymbol{M})$ 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 built (trained) for each word $\boldsymbol{O}, \boldsymbol{O} \in \boldsymbol{W}$, $\boldsymbol{W}$ is 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 $P(\boldsymbol{O} \mid \boldsymbol{M})$ 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 $P(\boldsymbol{O} \mid \boldsymbol{M}_{\mathrm{BW}})$, defined with (5), and first proposed by Welch [6], to reach the maximum in $\boldsymbol{O}$, i.e.

(6) $$\boldsymbol{P}_{\mathrm{BW}} = P(\boldsymbol{O} \mid \boldsymbol{M}_{\mathrm{BW}}) = \max_{\boldsymbol{M}} \{P(\boldsymbol{O} \mid \boldsymbol{M})\} .$$

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

(7) $$P^*(\boldsymbol{O} \mid \boldsymbol{M}) = \max_{\{X\}} P(\boldsymbol{O}, X \mid \boldsymbol{M}) \quad < \quad \sum_{\{X\}} P(\boldsymbol{O}, X \mid \boldsymbol{M}) = P(\boldsymbol{O} \mid \boldsymbol{M}) ,$$

and consequently:

(7a) $$\boldsymbol{P}_{\mathrm{Vi}} = P(\boldsymbol{O} \mid \boldsymbol{M}_{\mathrm{Vi}}) = \max_{\boldsymbol{M}} \left\{ P^*(\boldsymbol{O} \mid \boldsymbol{M}) \right\} \quad < \quad \boldsymbol{P}_{\mathrm{BW}} .$$

Here, instead of summation over all possible paths, cf. (1), $P(\boldsymbol{O} \mid \boldsymbol{M}_{\mathrm{Vi}})$ 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 $\alpha_i(t \mid \boldsymbol{M})$ and $\beta_i(t \mid \boldsymbol{M})$ by analogy to (4), (4a), what usually sharply improves the computational performance.

Viterbi's algorithm is very appropriate for the recognition regime, when HMM is enough trained. At the very beginning of the training and often during the 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 $P(\boldsymbol{O} \mid \boldsymbol{M})$ calculation [1, 6, 8], the output probabilities $b_i(O_t)$, i.e. the B matrix elements for a given state s_i of HMM, are modeled by a Q-dimensional Gaussian distribution of density $G(\mu_i, \Sigma_i)$, $i = 1 \div N$. Each centre μ_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 a diagonal one, i.e. that all cepstral vector components $o_t(q) \in O_t \in \boldsymbol{O}$, $q = 1 \div Q$, $t = 1 \div T$ are uncorrelated. Thus, each model $G(\mu_i, \Sigma_i)$ simplified to its $G(\mu_i, \sigma_i)$ can be represented as multiplication of 1D Gaussians, and independently examined by the respective outputs $b_i(O_t)$, $b_i(O_t) = b_i(o_t(q) \mid q = 1 \div Q)$ along the coordinates (q), $q = 1 \div Q$:

$$b_i(O_t) = G(O_t; \mu_i, \sigma_i) = \prod_{q=1}^{Q} G(o_t(q); \mu_i(q), \sigma_i(q)) =$$

$$= \frac{1}{\sqrt{(2\pi)^Q}} \prod_{q=1}^{Q} \frac{1}{\sigma_i(q)} \exp\left(- \frac{(o_t(q) - \mu_i(q))^2}{2\sigma_i^2(q)} \right), \tag{8}$$

$$\mu_i = (\mu_i(q) \mid q = 1 \div Q)\ ,\ \sigma_i = (\sigma_i(q) \mid q = 1 \div Q)\ ,\ i = 1 \div N,\ t = 1 \div T\ ,$$

where μ_i and σ_i are the respective vectors of centers and mean-square deviations.

Obviously, $P(\boldsymbol{O} \mid \boldsymbol{M})$ could overcome "very quickly" the lower limit of the computer numbers representation, e.g. ~10^{-308} for the C-language type "double". For instance, in accordance with (3), (4), (4a), (5) and (8), we can evaluate roughly for the output $P(\boldsymbol{O} \mid \boldsymbol{M})$:

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

The better a HMM is trained for given word, the "sharper" its Gaussians become which results in further more diminishing of the modeled probabilities. For this reason a computation scaling for the intermediate probabilities $\alpha_i(t)$ and $\beta_i(t)$ is usually proposed, [6]. In 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 of this could be the use of log-probabilities, where it is appropriate, but a 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 molds the input sequence $\boldsymbol{O} = (O_1, O_2, ..., O_T)$ by a combination of N in number Q-dimensional "Gaussians" each of them being just a multiplication of Q one dimensional (1D) Gaussians.

By definition, the density $G(o;\mu,\sigma)$, $\mu = \mu_i(q)$, $\sigma = \sigma_i(q)$, $i = 1 \div N$, $q = 1 \div Q$, of each 1D-Gaussian is a continuous function (pdf) over the input $o = o_i(q)$ and the parameters μ and σ. This leads to the necessity for 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 Gaussians considered.

As it is well known, the G-pdf as density is a differential of the respective distribution function, see also (10) and (10a) below. So the probability $P(x)$, modeled by given 1D-Gaussian $G(x) = G(x;\mu,\sigma)$ can be evaluated by the well known "trapezium-like" formulae:

$$P(x) = P(x - \Delta/2 \le X < x + \Delta/2) \approx \big(G(x - \Delta/2) + G(x + \Delta/2)\big)\frac{\Delta}{2}. \tag{9}$$

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 (Fig. 5), while $P(x)$ decreases itself that could lead to undesirable "underflow" in the computer calculations.

On the other hand, by a chosen sampling unit Δ the computation accuracy remains dependent only on the value of σ, 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 σ_0, $\sigma_0 = \sigma_0(\Delta)$, depending on the choice of Δ only, so that for each $\sigma, \sigma < \sigma_0$, the calculation of $P(x)$, e.g. by (9), will lose an accuracy. An illustration of this is given on Fig. 5.

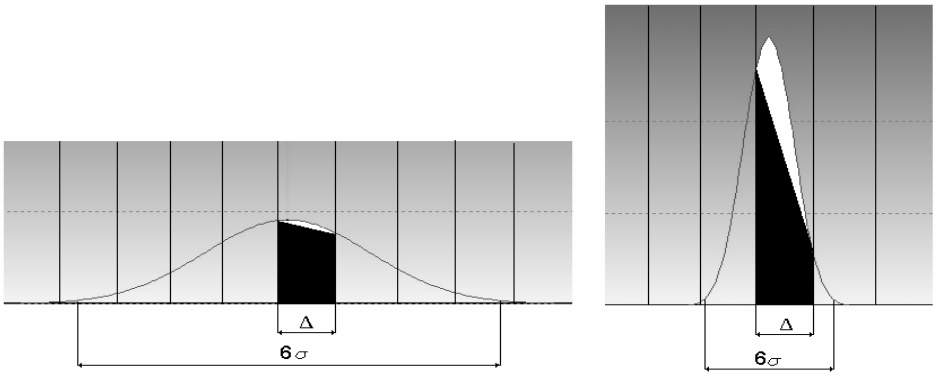


Fig. 5. Computational precision of probabilities depending on the choice of σ and Δ: (a) by $\sigma \ge \sigma_0(\Delta)$; (b) by $\sigma < \sigma_0(\Delta)$

So, the formula (9), we will call it *large G-model*, is used for 1D-Gaussians of $\sigma \geq \sigma_0$. In the cases of $\sigma < \sigma_0$, we introduce another calculation, called here a *slim G-model*, where the classical integration formulae is used:

$$(10) \qquad P(x) = \int_{x-\Delta/2}^{x+\Delta/2} G(u;\mu,\sigma)du = \Phi\left(\frac{x-\mu+\Delta/2}{\sigma}\right) - \Phi\left(\frac{x-\mu-\Delta/2}{\sigma}\right).$$

Here, $\Phi(x)$ is the well-known Laplace function that represents the cumulative Gaussian distribution:

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

and that could be tabulated only once at the beginning.

Intuitively, not bad evaluation of the threshold σ_0 for switching between the both models, (9) and (10) could be given with the choice:

$$(11) \qquad \Delta=1 \text{ and } \sigma_0 = 1.$$

We will try to evaluate more precisely the threshold $\sigma_0 = \sigma_0(\Delta)$ for an arbitrary Δ, $\Delta>0$.

4. Evaluation of the switching threshold between the Large and Slim G-models

The approximation (9) of the proposed large-G-model is based on the formula of "1 trapezium per Δ unit" type. A significantly more precise approximation $P_2(x)$ can be also given by the next formula – for "2 trapezia per Δ unit":

$$(12) \quad P_2(x) = P(x-\Delta/2 \leq X < x+\Delta/2) \approx \big(G(x-\Delta/2) + G(x) + G(x+\Delta/2)\big)\frac{\Delta}{4}.$$

Thus, the current error E_L of using the large-G-model can be estimated via the difference $P_2(x) - P_1(x)$, namely:

$$(12a) \quad E_L(x;\sigma,\Delta) = P_2(x) - P_1(x) = \big(G(x-\Delta/2;\sigma) + G(x+\Delta/2;\sigma) - 2G(x;\sigma)\big)\frac{\Delta}{4},$$

where for completeness we have set $P_1(x) = P(x)$ and $G(x;\sigma) = G(x)$, see (9).

In a similar way, we can define also the current error E_S in using the slim-G-model:

$$(13) \quad E_S(x;\sigma,\Delta) = \Phi_2\left(\frac{x+\Delta/2}{\sigma}\right) - \Phi_1\left(\frac{x+\Delta/2}{\sigma}\right) - \left(\Phi_2\left(\frac{x-\Delta/2}{\sigma}\right) - \Phi_1\left(\frac{x-\Delta/2}{\sigma}\right)\right),$$

where $\Phi_1(x)$ and $\Phi_2(x)$ denote two consecutive approximations of the Laplace function (10a) evaluated by the "1 trapezium" formula and "2 trapezia" formula correspondingly.

Now we can define the respective mean and/or maximal values for E_L and E_S, and search by them the optimal threshold σ_0 for switching between the models:

(14) $$E_{\text{mean}\,L}(\sigma,\Delta)=\sum_{x=-\infty}^{+\infty}\left|E_L(x;\sigma,\Delta)\right|P(x),\ E_{\max L}(\sigma,\Delta)=\max_{x\in(-\infty,+\infty)}\left\{\left|E_L(x;\sigma,\Delta)\right|\right\},$$

(15) $$E_{\text{mean}\,S}(\sigma,\Delta)=\sum_{x=-\infty}^{+\infty}\left|E_S(x;\sigma,\Delta)\right|P(x),\ E_{\max S}(\sigma,\Delta)=\max_{x\in(-\infty,+\infty)}\left\{\left|E_S(x;\sigma,\Delta)\right|\right\}.$$

It can be proven that our goal functions from (14) and (15) are relatively smooth either on σ or on Δ, and that they have most often 1 maximum (at least 2 maxima) for relatively small values of σ as well as they asymptotically converge to zero with increasing of σ (Fig. 6). Temporarily, we will assume that Δ = const.

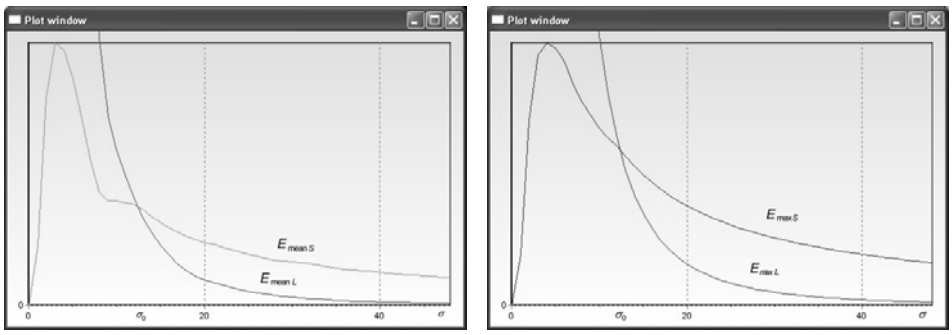
In brief, considering (14) and (15), we can evaluate two optimal values for the threshold σ_0 applying a criterion for a minimal error of both the defined G-models of joint implementation, either for the mean error

(16) $$E_{\text{mean}}(\sigma,\Delta)=\min\left\{E_{\text{mean}\,L}(\sigma,\Delta),E_{\text{mean}\,S}(\sigma,\Delta)\right\}\ \Rightarrow\ E_{\text{mean}\,L}(\sigma_0,\Delta)=E_{\text{mean}\,S}(\sigma_0,\Delta)$$

or for the maximal error

(17) $$E_{\max}(\sigma,\Delta)=\min\left\{E_{\max L}(\sigma,\Delta),E_{\max S}(\sigma,\Delta)\right\}\ \Rightarrow\ E_{\max L}(\sigma_0,\Delta)=E_{\max S}(\sigma_0,\Delta).$$

Algorithmically, the computation of σ_0, following both approaches (16) and (17), can be designed like searching of an unique (or eventually multiple) cross-point of the respective two graphics, see Fig. 6 and Table 1 (the row L=100).



(a) (b)

Fig. 6. Two estimations for the optimal threshold σ_0 by Δ=0.376:
(a) $\sigma_0=\sigma_0(\Delta)\,|\,(E_{\text{mean}L}(\sigma_0,\Delta)=E_{\text{mean}S}(\sigma_0,\Delta))$, and (b) $\sigma_0=\sigma_0(\Delta)\,|\,(E_{\max L}(\sigma_0,\Delta)=E_{\max S}(\sigma_0,\Delta))$

Table 1. Optimal threshold σ_0 evaluated for 4 cases of the sampling unit Δ.

L	Δ	$\frac{\sigma_0}{\Delta}$ by E_{mean}	Switching E_{mean} i.e. $E_{\text{mean}}(\sigma_0,\Delta)$	$\frac{\sigma_0}{\Delta}$ by $E_{\max}$	Switching $E_{\max}$ i.e. $E_{\max}(\sigma_0,\Delta)$
50	0.752	1.3338	0.0113892	1.3322	0.0163991
100	0.376	1.3295	0.0013606	1.3294	0.0024934
200	0.188	1.3294	0.0001767	1.3293	0.0003266
500	0.075	1.3293	0.0000114	1.3293	0.0000212

In Table 1, the obtained values of σ_0 are given for four different values of the sampling unit Δ, i.e. for the number L of sampling points of $P_1(x)$, $P_2(x)$, $\Phi_1(x)$, and $\Phi_2(x)$, cf. (9), (12) and (13).

The differences in the respective optimal thresholds σ_0 as dependant on Δ (i.e. on L) can be considered as being within the limits of precision determined by the corresponding approximation formula, namely: "1 trapezium per Δ" to estimate the G-model itself, and "2 trapezia per Δ" to estimate the error of using it. The differences in the evaluation of σ_0 either by E_{mean} or by $E_{\max}$ are also not very significant, i.e. they also obey the predetermined precision limits. Naturally, with an increment of L the preciseness of the above estimation becomes better.

In facts, the estimations (9) and (12) of the probabilities $P_1(x)$ and $P_2(x)$ as well as the evaluations (12a), (13), (14), (15), (16), and (17) of the approximation errors for the proposed switching between both G-models, the large (9) and the slim (10), should be considered independant on Δ, because of the obvious equations:

(18) $$G(x;\mu,\sigma,\Delta)=\frac{1}{\Delta}G\left(\frac{x}{\Delta};\frac{\mu}{\Delta},\frac{\sigma}{\Delta},1\right),\ \text{and}\ \Phi_{1(2)}\left(\frac{x\pm\Delta/2}{\sigma}\right)=\Phi_{1(2)}\left(\frac{x/\Delta\pm1/2}{\sigma/\Delta}\right).$$

Thus, a universal value can be generally adopted for the optimal σ_0 of switching between the both models, namely

(19) $$\sigma_0=\frac{\sigma_0(\Delta)}{\Delta}=\sigma_0(1)=1.3293\ =\ \text{const}\ \forall(\Delta>0).$$

Obviously (19) is a much better estimation than the initially supposed (11), in view of the larger errors for $\sigma\in(1,\sigma_0)$ if (11) is applied instead of (19), namely:

- for the min(E_{mean}) criterion (16): an extra error up to 82%, and
- for the min($E_{\max}$) criterion (17): an extra error up to 87%.

Additionally, we can set a new task for investigation supposing that (19) is the right (or at least very good) estimation for the threshold σ_0 between the large- and slim-G-models when using approximations of "*n* trapezia" type, *n*>1, instead of our case (*n* = 1).

5. An averaged HMM for given word versions

So far we have considered the training of HM for a 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 confidence. We have examined two possible approaches:

1. Instead of using a unique (Q-dimensional) Gaussian, each HMM state could be modeled by a mixture of Gaussians (Fig. 7), 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.

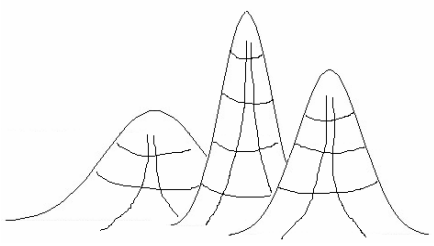


Fig. 7. A Gaussian mixture

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 after considering its performance effectiveness. Thus, each extra version of the given word is molded more simply, by training its own HMM only, followed by a final actualization of the respective averaged Gaussians. In this way we reach the idea of an averaged HMM, where the necessary restriction for a constant and a priori given number N of the HMM internal states seems very acceptable.

Let us have K versions less or more, distinguishing the input template (word). Let us also assume possible statistical dependences among the versions' cepstrums 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 a given coordinate (q), for the state s_i, i=1÷N, we have K number of 1D-Gaussians to average (Fig. 8).

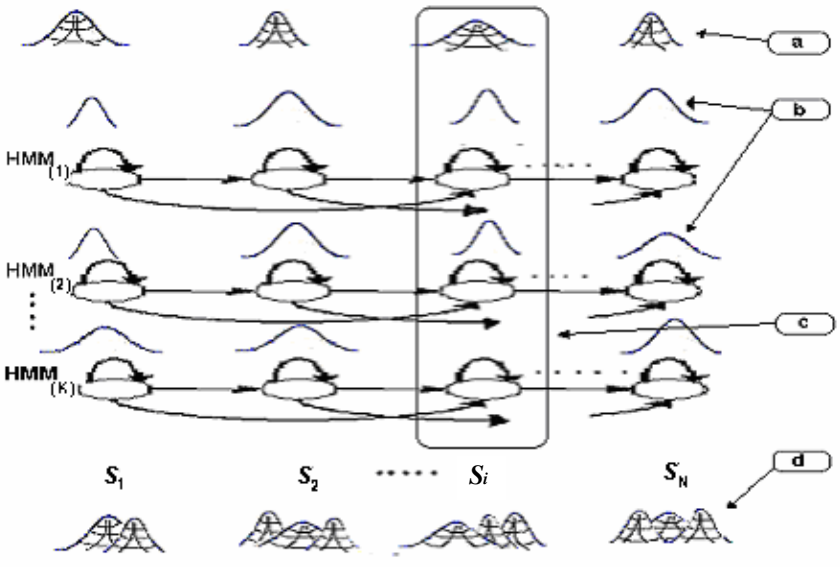


Fig. 8. HMM averaging: (a) Q-dimensional Gaussians for the states of $HMM_{(1)}$; (b) corresponding 1D-Gaussians over the axis (q), q=1÷Q for $HMM_{(1)}$ and $HMM_{(2)}$; (c) we average all 1D-Gaussians over the axis (q) for the state S_i of all $HMM_{(1\div K)}$; (d) mixture of Q-dimensional Gaussians for the states of the averaged $HMM_{(*)}$

As in Fig. 8, we will mark hereinafter the averaged HMM, its Gaussians and their attributes by an asterisk (*).

The centre $\vec{\mu}_{i,*}$ of the averaged Gaussian for the given state s_i can be evaluated as the mean vector:

$$\vec{\mu}_{i,*} = (\mu_{i,*}(q) \mid q = 1 \div Q) = \frac{1}{K}\sum_{k=1}^{K} \vec{\mu}_{i,k}\ ,\ i = 1 \div N, \tag{20}$$

$$\vec{\mu}_{i,k} = (\mu_{i,k}(q) \mid q = 1 \div Q),\ k = 1 \div K,\ i = 1 \div N\ ,$$

where $\vec{\mu}_{i,k}$ is the mean vector for the state s_i of the k-th HMM (i.e. for k-th word version).

For the vector $\vec{\sigma}_{i,*}$ of mean-square deviations of the averaged Gaussian for the given state s_i we can write

$$\vec{\sigma}_{i,*} = (\sigma_{i,*}(q) \mid q = 1 \div Q),\ i = 1 \div N\ , \tag{21}$$

where for its components $\sigma_{i,*}(q), q = 1 \div Q$, the following equation is always fulfilled:

$$\left(\sigma_{i,*}(q)\right)^2 = \frac{1}{K^2}\left(\sum_{k=1}^{K}\left(\sigma_{i,k}(q)\right)^2 + 2\sum_{1 \le k < l \le K} C_{i,(k,l)}(q)\right),\quad q = 1 \div Q,\ i = 1 \div N\ . \tag{21a}$$

Here $C_{i,(k,l)}(q)$ are the so called co-variation moments:

$$C_{i,(k,l)}(q) = \mathop{\mathrm{E}}_{t}\left(\left(o_{t,k}(q) - \mu_{i,k}(q)\right)\left(o_{t,l}(q) - \mu_{i,l}(q)\right)\right), q = 1 \div Q\ ,\ i = 1 \div N,\ k,l = 1 \div K\ ,$$

$\mathop{\mathrm{E}}_{t}(.)$ is the mean operator applied over the time t, t=1÷T, and for completeness only we could also write down that

$$\sigma_{i,k}^2(q) = \mathop{\mathrm{E}}_{t}[(o_{t,k}(q) - \mu_{i,k}(q))^2] = C_{i,(k,k)}(q)\ , q = 1 \div Q,\ i = 1 \div N,\ k = 1 \div K\ .$$

To escape the relatively difficult evaluation of the co-variation moments we should assume that the cepstral vector components $o_{t,k}(q) \in O_{t,k} \in \boldsymbol{O}$, $q = 1 \div Q$, $t = 1 \div T$, as numbered by k=1÷K, are uncorrelated. In this way we broaden the similar assumption of Section 2.5 over the input word versions (k), k=1÷K. As a result, all $C_{i,(k,l)}(q), k \ne l$, become equal to zero and equation (21a) – of the simpler type:

$$\sigma_{i,*}(q) = \frac{1}{K}\sqrt{\sum_{k=1}^{K}\left(\sigma_{i,k}(q)\right)^2}\ ,\ q = 1 \div Q,\ i = 1 \div N\ . \tag{21b}$$

Finally to meet the experimental results we extend (21b) to the following two inequalities:

$$\frac{1}{K}\sqrt{\sum_{k=1}^{K}\left(\sigma_{i,k}(q)\right)^2} \le \sigma_{i,*}(q) \le \frac{1}{K}\sum_{k=1}^{K}\sigma_{i,k}(q),\ q = 1 \div Q,\ i = 1 \div N\ , \tag{21c}$$

where the equality on the left is reached for independent items (1D-Gaussians), considering (21b). The equality on the right could also be reached probably for

some degree of dependency among the cepstral vector components $o_{t,k}(q)$, $t=1\div T$, $k=1\div K$, for each coordinate (q), $q=1\div Q$. Practically, (21c) gives us an interval for possible experimental variation of mean-square deviations of the averaged Q dimensional Gaussians, for each state s_i, i=1÷N–1, of the averaged HMM$_{(*)}$, cf. also Fig. 8.

As for the averaged transition matrix $A^* = \left\| a^*_{i,j} \right\|_{(N+1)\times(N+1)}$, it is simply calculated as the matrix sum

$$A^* = \frac{1}{K}\sum_{k=1}^{K} A(k)\,, \tag{22}$$

where $A(k) = \left\| a_{i,j}(k) \right\|_{(N+1)\times(N+1)}$ is the respective transition matrix A for the k-th HMM, $k=1\div K$. Equation (22) reflects the incompatibility assumption for corresponding probabilities $a_{i,j}(k)$, $k=1\div K$, for each couple (i,j), $i,j=1\div N$, i.e.

$$a^*_{i,j} = \frac{1}{K}\sum_{k=1}^{K} a_{i,j}(k),\ \ i,j=1\div N\ . \tag{22a}$$

Additionally, it is experimentally concluded that it is better the transition matrix averaging to be performed mean-geometrically instead of mean-arithmetically (22), because of the recognition improvement in this way.

6. Experimental results and analysis

The experiments are done on an IBM PC compatible: Intel Pentium4 3 GHz CPU, 1MB L2-cache, 512 MB RAM, 160 GB HDD. The operational system 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 method above proposed. By no special measures for optimizing the computing environment, the average training time for the whole database is about 5 min, i.e. about 2 s per word version. Experimental timing for 5 words of the database is given in the following Table 2.

Table 2. Experimental results for 5 sound words from Windows practice

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

Some fixed parameters concerning the experiments are: (a) the frame F length Δt for windowing the cepstrum computing, Δt = 10 ms; (b) the cepstral vectors' length Q = 40; (c) the HMM states' number, N = 7; (d) the maximal number of training iterations is 15.

Each word version is preliminary processed, i.e. the sound noise is cleared as better as possible and each word is isolated.

Under such conditions the average recognition rate is about 98%. For the present the automatic end-point detection is not precise enough 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 believe that the recognition rate is to become better if the averaged HMM is trained by more versions per word, what, of course will increase the training duration.

7. Conclusion and discussion

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. Two models are proposed, a large-G-model and a slim-G-model, and optimal threshold σ_0 is evaluated for switching between them depending on the current value of the Gaussian deviation σ, cf. (19). The method aims at providing the monotone convergence of the HMM trained by Baum-Welch and the consecutive recognition by Viterbi. Practically, at early stages of HMM training (when most often $\sigma > \sigma_0$) the large-G-model is recommended, while at the last stages of training as well as in recognition regime (i.e. when $\sigma < \sigma_0$ is expected) the slim-G-model should be preferred.

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

In near future we intend to develop an approach for automatic optimal choice of the HMM internal states number adaptively to each input word length and content. Besides, an effective (either fast or noise tolerance) content based access method to the DB of HMMs is also intended to help the recognition task by precedents.

The final purpose of the research work is to develop a HMM based method for speech recognition with automatic adapting to the specifics of Bulgarian language. A definite expectation in this respect is due to the experimental hypothesis of [7] for the almost full 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 Inform. Technology. 13 p.
   http://www.fit.vutbr.cz/~cernocky/oldspeech/lectures/hmm.pdf
2. Dunn, B. Speech Signal Processing and Speech Recognition, Current Topics in DSP. – In: Speech Proc. 2, RBD, May 13, 2003, p. 34.
   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, 264-288.
4. Jackson, P. Advanced Digital Signal Processing. UniS, Univ. of Surrey, p. 14.
   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, p. 9.
   http://www.cnel.ufl.edu/~yadu/report1.html
6. Rabiner, L. R. Tutorial on Hidden Markov Models and Selected Applications in Speech Recognition. – In: Proc. of the IEEE, Vol. **77**, 1989, No 2, 257-286.
7. Totkov, G. A. Conceptual and Computer Modelling of Language Structures and Processes (with Applications for the Bulgarian Language). DSc Dissertation, Spec. No 01-01-12 "Informatics". Univ. of Plovdiv, 2004 (Resume in Bulgarian. 57 p.).
8. Wu, Y. A. Ganapathiraju, 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. 2nd Ed. NY, John Wiley & Sons, Ltd., 2000, 227-261.